Contributions à l'apprentissage par renforcement inverse

Edouard Klein

Ecole Doctorale IAEM
Département Informatique

Thèse de Doctorat

Edouard Klein

Contributions à l'apprentissage par renforcement inverse

Thèse soutenue le 21 Novembre 2013

Membres du jury

Directeur de thèse	Dr. Yann Guermeur	Directeur de recherche, Equipe ABC, Loria-CNRS – Nancy
Co-directeur de thèse	Dr. Matthieu Geist	Enseignant-chercheur, Equipe IMS-MaLIS, Supélec – Metz
Rapporteurs	Dr. Rachid Alami	Directeur de recherche, LAAS / CNRS – Toulouse
	Prof. Brahim Chaib-draa	Professeur, DAMAS, Université Laval – Québec
Examinateurs	Prof. Sylvain Contassot-Vivier	Professeur, Loria-Université de Lorraine – Nancy
	Dr. Guillaume Laurent	Maître de conférences, Institut FEMTO-ST / ENSMM – Besançon
	Dr. Manuel Lopes	Chargé de recherche, Equipe Flowers, INRIA – Bordeaux
	Dr. Olivier Pietquin	Enseignant-chercheur, Equipe IMS-MaLIS / UMI 2958, Supélec – Metz

Equipe IMS-MaLIS
Supélec – Metz

Equipe ABC
Loria-CNRS – Nancy

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des théorèmes et propositions

Acronymes

PDM Processus Décisionnel de Markov. 17, 24–26, 40, 41, 44, 45, 47, 51–57, 63–67, 69, 70, 73, 77, 79, 81, 82, 87, 89, 90, 105

ARI Apprentissage par Renforcement Inverse. 3, 7, 11, 24–27, 29, 31, 35, 45–48, 51–57, 59, 63–67, 69, 71, 73–77, 79, 81, 83–85, 89–91, 100, 103–105

PD Programmation Dynamique. 11, 24, 25, 43, 46, 56, 60, 61, 76

AR Apprentissage par Renforcement. 3, 23–25, 29, 31, 43, 44, 46, 51–53, 55–57, 60, 61, 63, 66–68, 77, 89–91, 99, 105

LSPI *Least Squares Policy Iteration*. 11, 24, 25, 44, 45, 64–66, 74, 77, 90, 99

PIRL *Projection Inverse Reinforcement Learning*. 53, 54

MMP *Maximum Margin Planning*. 54, 70

PM *Policy Matching*. 53

MWAL *Multiplicative Weights for Apprenticeship Learning*. 54

MaxEnt *Maximum Entropy*. 55, 57

RelEnt *Relative Entropy*. 25–27, 57, 65, 67, 73–77, 81, 83, 84, 90, 103

LPAL *Linear Programming for Apprenticeship Learning*. 54

GPIRL *Gaussian Processes Inverse Reinforcement Learning*. 56

LSTD-μ *Least Squares Temporal Differences feature expectations*. 4, 7, 11, 25, 26, 61, 62, 64–66, 69–72, 76, 77, 90, 103, 104

LSTD *Least Squares Temporal Differences*. 11, 25, 26, 43, 44, 46, 61, 64, 66, 90

SCIRL *Structured Classification for Inverse Reinforcement Learning*. 4, 9, 11, 13, 24, 26, 27, 63, 68–77, 79, 81–87, 90, 91, 93, 104, 105

CSI *Cascaded Supervised learning for Inverse reinforcement learning*. 4, 5, 9, 11, 13, 24, 26, 27, 68, 71, 79–87, 90, 91, 95, 104

CNN *Convolutional Neural Network*. 37

SVM *Support Vector Machine* (Machine à Vecteurs Support). 24, 35, 36, 74, 76, 91

GMM *Gaussian Mixture Model* (Modèle à base de Mélange de Gaussiennes). 37

ALVINN *Autonomous Land Vehicle In a Neural Network*. 37

CHURPs *Compressed Heuristic Universal Reaction Planners*. 38

k-NN *k-plus proches voisins* (en anglais *k-Nearest Neighboors*). 39

PG Processus Gaussiens. 25, 55, 56

SMILe Stochastic Mixing Iterative Learning. 38

DAGGER Dataset Aggregation. 38

Notations

d_ψ Dimension de l'espace d'attributs sur l'espace d'état. 31, 32

k Entier indexant les dimensions d'un espace. 32

$\mathfrak{m}$ Centre d'une gaussienne. 32

C_f Coefficient de concentration dans le pire des cas. 71, 94

λ Coefficient de régularisation. 44, 62

q Fonction de score pour la classification. 31, 32, 34, 35, 68, 69, 71, 72, 79, 80, 82, 83, 94, 96

μ^E Attribut moyen de l'expert. 54, 57, 64, 65, 69–72

∇ Sous-gradient d'une fonction. 31, 35, 57, 70, 71

$\mathbb{R}$ Le corps des réels. 25, 31, 34, 41, 42, 47, 52, 59, 99

$d_\mathcal{S}$ Dimension de l'espace d'état. 32

$\mathfrak{l}$ Fonction de marge dans le classifieur à marge. 34, 35, 70, 71

π Une politique. 11, 19, 29, 38, 40–44, 47, 48, 54, 59–64, 69, 72, 95

$\hat{R}^C$ Approximation de R^C. 80, 82, 83, 85, 95–97

s État. 29, 30, 32, 34–36, 40–48, 51, 52, 57, 59–62, 68–72, 79–82, 85, 96

N Nombre d'exemples dans une base. 30, 34, 35, 44, 61–64, 75, 80–82, 85

ρ Loi de probabilité ou fonction de poids. 19, 30, 31, 46

ϕ Fonction d'attributs état-action. 31, 32, 34, 35, 43, 44, 46, 47, 57, 59–62, 64, 65, 69–71, 85

a Une action. 30, 32, 34, 35, 40–44, 46, 47, 51, 52, 59–62, 68–72, 79–82, 85

u Règle de mise à jour pour les algorithme d'ARI itératifs. 54, 64

J Fonction de coût de la classification structurée. 35, 70, 71

j Entier indexant les dimensions d'un espace. 32, 57, 60–63, 85

$\hat{r}$ Echantillon pour l'estimation de R^C. 80–82, 85

θ Vecteur de paramètres pour l'ARI. 46–48, 53, 54, 57, 61, 64, 69–73, 93–95

L Longueur d'une trajectoire. 57, 59, 60, 65, 70

ρ_E Distribution stationnaire de l'expert. 38, 41, 46, 71, 72, 81–83, 93–97

ζ Matrice de paramètres pour LSTD-μ. 61, 62, 65

γ Facteur d'amortissement. 41, 42, 44, 47, 52, 57, 59–62, 69, 70, 74, 79, 80, 83, 93, 95, 96

R^C Fonction de récompense induite par la fonction de score d'un classifieur. 80, 82, 83, 95–97

Φ Espace d'attributs état-action. 31, 34, 36, 47

M Nombre de trajectoires. 57, 59, 60, 65

ϵ_C^{emp} Erreur empirique de classification. 30

$\mathcal{A}$ Espace d'action. 29, 31, 32, 34, 35, 40–44, 51, 52, 59, 68, 70–72, 79, 80, 83, 94

ζ Variable d'écart. 34, 35

π^C Politique issue d'un classifieur. 30, 34, 35, 38, 68, 71, 72, 80–83, 94–96

κ Noyau pour une SVM. 36

R Fonction de récompense. 19, 41–48, 52, 54, 57, 59–64, 69–73, 93–95

$\mathcal{M}$ Un PDM. 41

t Indice temporel. 38, 40, 41, 47, 57, 59–61, 69, 70

d Distance entre deux attributs moyens. 54, 64

ξ Vecteur de paramètres pour l'AR. 43, 44

d_ϕ Dimension de l'espace d'attributs état-action. 31, 85

$\mathcal{S}$ Espace d'état. 29–32, 34, 36, 40–43, 47, 48, 51, 52, 59, 68, 71, 72, 79, 80, 82, 94, 96

R^E Récompense optimisée par l'expert. 45, 46, 68, 74, 79, 83

π^E Politique de l'expert. 30, 35, 44–46, 54, 57, 63–65, 67–72, 75, 79–83, 93–97

i Indice des exemples dans une base. 30, 34, 35, 44, 45, 60, 64, 70, 71, 80–82, 85

σ Ecart-type d'une gaussienne. 32

m Entier indexant un ensemble. 57

ω Vecteur de paramètres pour la classification. 31, 32, 34, 35

D_{sar}^q Base d'entraînement du régresseur dans CSI. 80–82, 85

p Probabilités de transition. 40, 41

ψ Fonction d'attribut sur l'espace d'état. 31, 32, 36, 47, 59

$\mathbb{1}$ Fonction indicatrice. 30, 32, 71, 81, 94, 96

x^* Element issu d'un $\arg\max_x$. 35, 70

D_{sa}^π Base de type s, a obtenue en suivant la politique π. 30, 35, 44, 60, 75, 80

$\mathbb{E}\left[f(x) | x \sim \rho\right]$ Espérance de $f(x)$ pour x tiré selon ρ. 30, 41, 46, 47, 59, 69, 71, 72, 81, 82, 93–95

d_f Dimension de l'espace engendré par f. 59–63, 71

D_{sas}^π Base de type s, a, s' obtenue en suivant la politique π. 44, 57, 63–65, 67, 69–71, 74, 75, 80–82

B_R^* Opérateur d'optimalité de Bellman. 43, 93, 94

$||\cdot||_x$ Norme x. 47, 48

X^T Transposée de la matrice X. 31, 32, 34, 35, 41, 43, 44, 46–48, 57, 61, 62, 69–72, 82, 85, 94–97

$\mathfrak{G}_\sigma^m$ Fonction gaussienne de centre m et d'écart-type σ. 32

μ^π Attribut moyen de la politique π. 11, 47, 48, 59–63

g_i Nombre de gaussiennes pour la dimension i dans un vecteur d'attributs basé sur un réseau de gaussiennes. 32

$\hat{X}$ Approximation de X à partir de données. 46, 47, 54, 64

P^π Matrice des probabilités de transition induites par la politique π. 40–42, 60, 61, 71, 82, 93–96

ρ_π Distribution stationnaire induite par la politique π. 41

$p(s'|s,a)$ Probabilité qu'un agent transite en s' après avoir choisi l'action a dans l'état s. 40–42, 51, 61, 79, 80, 95, 96

π^*_R Une politique optimale pour la fonction de récompense R. 42, 43, 46, 82, 83, 93, 95–97

B^A Ensemble des applications de A dans B. 29–31, 34, 40–42, 52, 59, 71, 72

$(f(i,j))_{i,j}$ Matrice dont l'élément de ligne i et de colonne j est $f(i,j)$. 40, 85

X_j Composante j du vecteur X. 32, 57, 60–62, 85

C_π Coefficient de concentration entre une politique et la politique experte. 82, 83, 95, 96

$|A|$ Cardinal de l'ensemble A. 31, 32, 40, 72

$\langle x,y\rangle$ Produit scalaire de x et y. 36

Q^π_R Fonction de qualité de la politique π pour la récompense R. 42, 43, 68, 69, 72, 79, 94

V^π_R Fonction de valeur pour la récompense R lorsqu'on suit la politique π. 41–43, 47, 48, 60–63, 82, 95–97

B^π_R Opérateur d'évaluation de Bellman. 42–44, 93–95

$g(\pi)$ Politique gloutonne vis-à-vis de la fonction de qualité de π. 42–44, 72

ϵ^ρ_C Risque de classification sur la distribution ρ. 30, 46, 71, 72, 81, 82, 94, 95

D^R_{sasr} Base de type $s,a,s\prime,r$. 44, 45, 62, 64

Anglicismes

batch Par paquet. 26, 45, 61, 73, 90

off-policy Signifie que la politique qui contrôle le système n'est pas celle qui est évaluée. 26, 45, 46, 53, 61, 63, 70, 73, 90

on-policy À la différence du *off-policy*, la politique évaluée est celle qui contrôle le système. 26, 53, 70

reward shaping Transformation de la récompense ne changeant pas les politiques optimales. 3, 25, 51

boosting Augmentation du pouvoir descriptif de la représentation. 36, 37, 56

mountain-car Problème jouet où une voiture doit sortir d'une vallée. 4, 7, 9, 26, 32, 33, 49, 73–76, 83, 84, 99–101

highway Problème jouet où une voiture doit doubler d'autres voitures sur une autoroute. 4, 7, 9, 26, 73, 75, 76, 83–86, 100, 101

1
Introduction

Ce chapitre expose les motivations de nos travaux et situe nos contributions. Nous regroupons la liste de ces contributions ainsi que des publications auxquelles elles ont donné lieu en annexe. Nous terminons ce chapitre par une vue d'ensemble du manuscrit.

1.1 Motivations

Dans les pays développés, peu de travaux confiés à des humains peuvent aujourd'hui être confiés à des machines [1]. Les limitations des systèmes automatiques ne sont pas mécaniques, les machines aujourd'hui peuvent être plus puissantes et robustes que n'importe quel animal et plus précises et plus fiables qu'un bon artisan. La généralisation de l'automatisation est freinée par l'impossibilité pour une machine d'assumer la charge cognitive liée à la gestion des environnements dynamiques et incertains dans lesquels les humains parviennent à évoluer.

1. Dans les pays en développement ou non développés, il est plus rentable de payer un travailleur non qualifié que d'utiliser un robot ou un ordinateur.

Le problème du *contrôle optimal* consiste à influencer le comportement d'un système dynamique pour qu'il remplisse une tâche. Le concepteur d'un système informatique résolvant le problème du contrôle optimal doit faire preuve d'une double expertise :

- il doit comprendre comment réaliser la tâche, c'est-à-dire prendre en compte les critères de réussite et d'échec, les risques associés à un échec catastrophique, les compromis entre les différents coûts (en temps, en énergie, en matières premières etc.) ;
- il doit également savoir exprimer ces contraintes dans le cadre d'une méthode de recherche du contrôle optimal.

Pour des tâches complexes il est coûteux de réunir ces deux types d'expertise dans une même équipe de développement. Pour réduire l'expertise nécessaire dans le domaine du contrôle optimal, nous allons, à partir d'une *démonstration* du contrôle, extraire automatiquement une *description des contraintes de la tâche* exploitable par des méthodes de recherche du contrôle optimal. Ce paradigme, l'*apprentissage par imitation*, vise à permettre à un utilisateur non qualifié dans le domaine informatique, mais performant en ce qui concerne la tâche à effectuer, d'enseigner de nouveaux comportements à une machine simplement en en faisant l'exemple.

L'application de ce paradigme à des domaines dynamiques dont le modèle est inconnu et possiblement non linéaire nous pousse à choisir l'Apprentissage par Renforcement (AR) comme méthode de recherche du contrôle optimal. L'apprentissage par renforcement permet de manière générique [2] d'apprendre le contrôle optimal à partir d'un signal de récompense, information locale sur la qualité du contrôle, en exploitant l'information issue d'interactions avec le système. Nos contributions se situent donc

2. C'est-à-dire qu'un même algorithme peut être utilisé pour apprendre à gérer deux domaines complètement différents.

dans le domaine de l'Apprentissage par Renforcement Inverse (ARI), qui cherche à obtenir un signal de récompense cohérent avec une démonstration de la tâche que fournit un expert.

1.2 *Vue d'ensemble du manuscrit*

Le **chapitre 2** explore la problématique de l'apprentissage par imitation. Nous commençons par formaliser la notion de politique déterministe markovienne : à un état contenant toutes les données nécessaires à la prise d'une décision, l'agent associe de manière déterministe une action. Nous remarquons alors que le problème d'apprentissage supervisé d'une politique se réduit au problème bien connu de la classification multi-classe, les états étant les entrées à classifier et les actions étant les catégories à associer aux entrées. Nous définissons le risque empirique et le risque réel, qui mesurent l'erreur de classification respectivement sur les données dont nous disposons et sur la distribution sur laquelle il nous importe d'apprendre la politique experte. Nous nous attardons ensuite sur les modèles linéaires, de grande importance pour nous car nous les utiliserons tout au long du manuscrit comme modèle d'approximation des fonctions de valeur, de score, de récompense et d'attribut moyen de l'expert. Nous proposons deux choix de fonction d'attributs génériques, que nous utiliserons dans nos expériences. Ces prolégomènes nous amènent à étudier plus particulièrement deux méthodes de classification, une méthode à marge structurée que nous retrouverons au chapitre 5 lors de l'exposé de l'algorithme *Structured Classification for Inverse Reinforcement Learning* (SCIRL) et les *Support Vector Machine* (Machine à Vecteurs Support) (SVM), qui nous serons utiles au chapitre 6 comme routine de l'algorithme *Cascaded Supervised learning for Inverse reinforcement learning* (CSI).

Après un état de l'art des méthodes d'imitation supervisée, nous définirons le cadre des Processus Décisionnel de Markov (PDM) dans lequel se situera le reste du manuscrit. La notion de dynamique temporelle, formalisée comme les probabilités de transition vers un état d'arrivée d'un agent choisissant une certaine action dans un état de départ, y est fondamentale. Elle permet de dépasser le cadre de la politique comme simple association état-action que manipulent les méthodes supervisées en introduisant des critères long terme sur la qualité d'une politique. La fonction de valeur, espérance de la somme pondérée d'une fonction de récompense (signal localisé sur la qualité du contrôle) permet d'ordonner les politiques. La Programmation Dynamique (PD) est utilisée pour rechercher une politique optimale via la résolution des équations de Bellman. Nous nous intéresserons plutôt à l'AR, qui résout le même problème sans connaître explicitement les probabilités de transition, mais en interagissant avec le système à contrôler. D'un intérêt particulier pour nous sont les algorithmes pouvant fonctionner à partir d'une base de données recueillies une fois pour toutes, nous donnons l'exemple de *Fitted-Q* (algorithme 2) et *Least Squares Policy Iteration* (LSPI) (algorithme 3), plus efficaces en échantillons que les algorithmes en ligne.

L'AR permet de trouver une politique optimale à partir d'information localisée sur la qualité du contrôle présentée sous la forme d'une fonction de récompense. Le problème inverse, l'ARI, consiste à trouver une fonction de récompense pour laquelle un comportement dont nous observons l'exemple est optimal. L'optimisation de cette fonction par un algorithme d'AR fournira une politique résolvant le problème de l'apprentissage par imitation. Nous fournissons deux critères de résolution du problème de l'ARI (formules 2.51 et 2.52). L'un mesure l'optimalité de la politique d'imitation pour la récompense (normalement inconnue) optimisée par l'expert, l'autre mesure l'optimalité de l'expert

pour la récompense trouvée par l'algorithme d'ARI. Enfin, nous terminerons le chapitre par la définition de l'attribut moyen. L'existence de cette grandeur découle du choix d'une modélisation linéaire de la fonction de récompense. Elle joue un rôle capital dans la plupart des approches de l'état de l'art, car deux politiques d'attributs moyens proches auront (pour toute récompense) des valeurs proches (Proposition 1).

Le **chapitre 3** propose un état de l'art de l'ARI. Nous justifions tout d'abord notre choix de considérer la fonction de récompense comme une fonction de l'espace joint état-action dans $\mathbb{R}$, puis étudions les transformations qu'il est possible d'appliquer à une fonction de récompense sans modifier les politiques optimales associées (*reward shaping*). Nous faisons ensuite l'état de l'art de l'ARI proprement dit. La grande majorité des approches obéit à une structure commune que nous exposons algorithme 4. Dans cette approche itérative, la fonction de récompense est approximée par un modèle linéaire. L'attribut moyen de la politique optimale pour la récompense courante est comparé à l'attribut moyen de l'expert. Le vecteur de paramètres de la fonction de récompense est ensuite mis à jour afin de faire diminuer une certaine distance entre ces deux attributs moyens. Les algorithmes suivant ce schéma diffèrent par la définition de la distance entre deux attributs moyens et par la règle de mise à jour. Toutes ces approchent souffrent des problèmes intrinsèquement liés à leur structure itérative. Il faut de manière répétée résoudre un PDM, c'est-à-dire trouver la politique optimale pour une récompense arbitraire. Il faut ensuite estimer l'attribut moyen de cette politique optimale. Ces deux étapes sont gourmandes en données et en temps de calcul. Plus récemment, de nouvelles approches ce sont détachées de cette structure itérative. Certaines abandonnent également l'approximation linéaire de la fonction de récompense, la remplaçant par exemple par des Processus Gaussiens (PG). La plupart de ces nouvelles approches nécessitent cependant pour fonctionner au moins autant d'information que les approches itératives précédentes. L'une d'entre elle cependant, *Relative Entropy* (RelEnt) (algorithme 5) est particulièrement peu gourmande en données. Elle ne réclame dans sa version échantillonnée qu'une base d'exemples de l'expert et des échantillons tirés en application d'une politique aléatoire, couvrant tout l'espace d'état. De ce fait, elle nous servira de point de comparaison aux chapitres 5 et 6 lorsque nous testerons nos nouveaux algorithmes d'ARI sur des problèmes jouet.

Le **chapitre 4** décrit notre première contribution, l'algorithme *Least Squares Temporal Differences feature expectations* (LSTD-μ). Nous commençons par rappeler la définition de l'attribut moyen et constatons que, composante par composante, l'attribut moyen est une fonction de valeur (avec pour récompense la composante correspondante du vecteur d'attribut). Pour remplacer la méthode usuelle de calcul de l'attribut moyen (une estimation de Monte-Carlo, algorithme 6) nous nous proposons d'exploiter le constat que nous venons de faire, non pas en utilisant les outils de la PD (algorithme 7), qui nécessitent de connaître les probabilités de transition, mais en adaptant un algorithme d'estimation de la fonction de valeur bien connu en AR, *Least Squares Temporal Differences* (LSTD). L'algorithme obtenu, LSTD-μ, est une généralisation vectorielle de LSTD-Q. La complexité algorithmique asymptotique des deux algorithmes est la même car l'opération coûteuse, une inversion de matrice, n'a besoin d'être effectuée qu'une seule fois pour toutes les composantes de l'attribut moyen. Nous montrons comment les résultats théoriques disponibles sur LSTD peuvent être adaptés à LSTD-μ. Nous plaçons ensuite cette contribution dans le cadre des approches d'ARI itératives dont la structure a été isolée au chapitre précédent. Les algorithmes d'AR tels LSPI permettant de résoudre un PDM grâce à une base de données recueillie à l'avance, nous pouvons maintenant grâce à LSTD-μ, qui peut estimer l'attribut moyen d'une politique arbitraire grâce à cette

même base de données, faire fonctionner ce type d'algorithmes d'ARI de manière *batch* et *off-policy*. Ce mode de fonctionnement nous permet d'espérer les appliquer à des problèmes où le modèle dynamique du système n'est pas disponible. Nous validons empiriquement notre démarche sur le problème jouet du pendule inversé, et constatons que l'estimation fournie par LSTD-μ permet à un algorithme d'ARI itératif d'atteindre les mêmes performances que lorsqu'il utilise l'estimation de l'attribut moyen fournie par un oracle ayant accès à un simulateur du système. La source d'erreurs semble donc se situer au niveau de la résolution du PDM à chaque itération. Il apparaît important de définir des algorithmes d'ARI ne nécessitant pas une telle résolution répétée.

Nous proposons un tel algorithme au **chapitre 5**. Rappelant les notions vues aux chapitres 2 et 3, nous établissons un parallèle entre la fonction de score d'un classifieur et la fonction de qualité de l'expert. Pour trouver l'action (l'étiquette) à associer à un état (une entrée), les classifieurs à fonction de score choisissent l'action qui associée à l'état obtient le score le plus haut. L'expert mettant en œuvre un politique optimale, il va utiliser le même mécanisme glouton de sélection d'action, non pas sur une fonction de score mais sur la fonction de qualité de sa politique. Grâce à la modélisation linéaire de la fonction de récompense (objet de la recherche menée par les algorithmes d'ARI), cette fonction de qualité peut s'exprimer comme le produit scalaire du vecteur de paramètres de la récompense et de l'attribut moyen de l'expert. Il est dont naturel d'utiliser également pour la fonction de score du classifieur une paramétrisation linéaire mettant en jeu l'attribut moyen de l'expert. Le principe de SCIRL, le nouvel algorithme d'ARI que nous proposons, est simplement d'apprendre directement la fonction de récompense grâce à un classifieur à fonction de score linéaire. Il suffit d'utiliser l'attribut moyen de l'expert dans la paramétrisation linéaire. L'algorithme de classification structurée étudié au début du manuscrit se prête bien à cette tâche. Nous proposons de plus une heuristique qui, en conjonction avec une méthode *on-policy* d'estimation de l'attribut moyen de l'expert (comme par exemple l'algorithme LSTD-μ du chapitre précédent [3]), permet d'utiliser SCIRL uniquement à partir de données échantillonnées par l'expert, ce qu'à notre connaissance aucun autre algorithme d'ARI ne peut faire sur un problème non trivial.

3. LSTD-μ peut en effet fonctionner, comme LSTD, au choix en mode *on-policy* ou *off-policy*.

Nous fournissons Théorème 1 une borne théorique sur les performances de SCIRL. Cette borne garantit que lorsque l'erreur d'estimation de l'attribut moyen de l'expert et l'erreur de classification sont faibles, alors la politique de l'expert est quasi-optimale pour la récompense trouvée par SCIRL, ce qui est un des critères de réussite exprimés au chapitre 2. Le Corollaire 1 laisse espérer que SCIRL évite les solutions dégénérées au problème de l'ARI. Les performances de SCIRL vis-à-vis du second critère de réussite du chapitre 2 font l'objet d'une étude empirique. Notre instanciation de SCIRL, avec l'algorithme de classification structurée discuté plus tôt en conjonction avec l'heuristique permettant le fonctionnement à partir de données expertes seules, est testée sur les problèmes jouet du *mountain-car* et du *highway*. Sur le premier, SCIRL obtient des résultats significativement meilleurs que ceux de RelEnt et ceux d'un algorithme d'imitation supervisée. Sur le second, les deux approches d'ARI montrent de meilleures performances que la classification, et SCIRL reste légèrement meilleur que RelEnt.

Basé sur le même constat de proximité entre fonction de score d'un classifieur et la fonction de qualité de l'expert, l'algorithme CSI fait l'objet du **chapitre 6**. Plutôt que d'introduire la structure temporelle du PDM par le biais de l'attribut moyen de l'expert, nous utilisons une version échantillonnée de l'équation de Bellman, où la fonction de qualité de l'expert est remplacée par la fonction de score d'un classifieur entraîné sur les données expertes. La classification intervient ici comme sous-routine. Elle est

la première étape d'une cascade. La seconde est une étape de régression où l'équation de Bellman inversée permet d'obtenir des données grâce auxquelles un régresseur apprend la fonction de récompense. Une nouvelle heuristique est proposée. En utilisant cette heuristique il est possible de définir les données fournies à l'étape de régression uniquement à partir de données expertes. Nous reproduisons Théorème 2 les résultats théoriques obtenus par un autre doctorant de l'équipe MaLIS. Si l'erreur de classification et l'erreur de régression sont faibles, alors la politique de l'expert est quasi-optimale pour la récompense trouvée par CSI. L'autre critère, à savoir l'optimalité vis-à-vis de la récompense de l'expert[4] de l'agent entraîné sur la récompense trouvée par CSI est là aussi l'objet d'une étude expérimentale sur les mêmes problèmes jouet que pour SCIRL. La classification s'avérant être moins performante que les méthodes d'ARI, nous ne gardons dans cette étude que RelEnt comme point de comparaison. Les performances de CSI sont sensiblement équivalentes à celles de SCIRL, un test d'égalité des moyennes montre qu'elles sont par endroit légèrement supérieures.

4. Normalement inconnue, mais nous travaillons sur des problèmes jouet.

Le **chapitre 7** résume les contributions apportées et propose des perspectives de recherche, l'étude de certaines ayant été entamée sans avoir pu être menée à terme dans le cadre de cette thèse. Il serait notamment utile de renforcer l'étude empirique de nos deux nouveaux algorithmes d'ARI en les appliquant à des problèmes réels. Les difficultés rencontrées lors des travaux préliminaires en ce sens (voir la section C.4) sont liées à l'optimisation de la récompense trouvée par les algorithmes d'ARI et non à la recherche de la récompense elle-même. Une solution à ce problème serait le développement d'algorithmes d'ARI fournissant une politique de contrôle et non simplement une récompense.

Les trois annexes regroupent pour l'**annexe A** la preuve des principaux résultats théoriques, pour l'**annexe B** une description des problèmes jouet sur lesquels nous évaluons empiriquement les nouvelles méthodes que nous proposons et pour l'**annexe C** une liste des publications et communications auxquelles cette thèse a donné lieu.

2
Apprentissage par imitation

L'ARI permet de transformer l'apprentissage de tâches complexes à partir d'un signal de récompense en apprentissage par imitation, en déduisant de démonstrations d'un expert un signal de récompense cohérent avec ces démonstrations. Ce chapitre formalise les notions qui nous seront utiles tout au long du manuscrit. Nous abordons l'apprentissage par imitation par le biais de l'apprentissage supervisé (discrimination). Ces méthodes ne passent pas par une fonction de récompense, mais apprennent directement l'association que fait l'expert entre entrées sensorielles et actions sur l'environnement. Nous en présentons un état de l'art. Nous précisons ensuite le cadre de l'AR, puis de l'ARI, dont nous ferons l'état de l'art au chapitre 3.

2.1 Imitation supervisée

Nous réduisons dans cette section le problème de l'imitation supervisée à celui de la classification. Après une étude rapide des notions centrales d'attribut et de généralisation, nous étudions deux cadres de classification que nous retrouverons aux chapitres 5 et 6 lorsque nous présenterons nos deux nouveaux algorithmes d'ARI. Nous terminons par un état de l'art de l'imitation supervisée.

2.1.1 Formalisme

Un agent, qu'il soit artificiel ou humain, qu'il s'agisse de l'expert, qu'il soit en cours d'apprentissage ou même qu'il agisse de manière aléatoire (utile pour l'exploration), met en œuvre une politique. D'un point de vue mathématique, une politique π est formellement définie comme une application d'un espace d'état vers un espace d'action [1] :

$$\pi \in \mathcal{A}^{\mathcal{S}}. \tag{2.1}$$

Cette politique encode le comportement de l'agent : dans un état $s \in \mathcal{S}$, l'agent choisira l'action $\pi(s) \in \mathcal{A}$. On constate que ce formalisme implique que pour choisir son action, l'agent n'utilise que les informations contenues dans l'état. Il faudra donc en pratique veiller à ce que celui-ci contienne toutes les informations utiles à la prise de décision, c'est-à-dire par exemple pour un système physique, non seulement les valeurs courantes des capteurs, mais aussi peut-être certaines valeurs passées afin de pouvoir calculer des taux de variation.

Les espaces d'état et d'action ne sont généralement pas des espaces vectoriels : la plage de valeurs que peut prendre une composante est rarement illimitée. Il s'agit même parfois d'ensembles finis. Dans ce cas, on s'autorisera à manipuler les états comme les $|\mathcal{S}|$ premiers entiers, puisqu'il existe un isomorphisme entre $\mathcal{S}$ et $[\![1; |\mathcal{S}|]\!]$. C'est dans les

1. Une autre manière de formaliser les choses repose sur les politiques stochastiques définies dans $[0,1]^{\mathcal{S}\times\mathcal{A}}$ (ignorons le formalisme des espaces probabilisés pour cette discussion). Cela complexifie un peu l'analyse, et la perte de généralité lorsque l'on se cantonne aux politiques déterministes est minimale (dans un cadre markovien).

problèmes qui nous intéressent quasiment toujours le cas pour l'espace d'action. Nous considérons par défaut qu'il s'agit d'un ensemble fini de petit cardinal. La plupart des notations que nous utiliserons supposent un espace d'état également fini, mais l'usage d'attributs judicieusement choisis (sous-section 2.1.3) permet dans la plupart des cas l'extension pratique au cas continu.

2.1.2 *Classification*

Il est possible de voir le problème de l'imitation comme celui de la recherche d'une politique correspondant à celle de l'expert, si possible pour chaque état ou au mieux en fonction des contraintes en mémoire et en temps de l'agent. Il est en effet évident que si les politiques des deux agents sont identiques alors ces deux agents accomplissent la même tâche avec le même degré d'efficacité. Même lorsque la politique de l'expert (notée π^E) est intégralement connue, cette formulation n'est pas forcément dénuée d'intérêt, il peut en effet être souhaitable de remplacer l'expert par un agent moins coûteux mais probablement plus limité. Apprendre parfaitement (ou apprendre au mieux avec quelques erreurs) la politique de l'expert est alors sensé.

Bien souvent, cependant, il est impossible de connaître intégralement la politique de l'expert, ne fut-ce que parce que l'espace d'état est trop grand. Il faut alors se contenter d'exemples sur un certain nombre N d'états[2] :

$$D_{sa}^{\pi^E} = \{(s_i, a_i = \pi^E(s_i)) | i \in [\![1; N]\!]\}. \tag{2.2}$$

2. L'indice sa de $D_{sa}^{\pi^E}$ indique quels éléments (ici des couples (s_i, a_i)) composent la base. A l'Équation 2.50, par exemple, ce sont des quadruplets (s_i, a_i, s_i', r_i) qui sont utilisés.

Le problème de l'imitation se trouve ainsi réduit à celui de la classification. Étant donné que l'espace d'action est fini et de petit cardinal, chaque action est vue comme une étiquette à appliquer à un état. La démonstration de l'expert fournit la base d'entraînement.

La classification est un problème plus subtil que celui de naïvement apprendre par cœur la base d'entraînement. Ce que nous cherchons à optimiser n'est pas la performance sur la base d'entraînement fournie, mais la performance sur l'espace d'état en général. Plus précisément, certains états nous intéressent plus que d'autres. Pour une justification intuitive, il suffit de penser aux jeux de plateau, où bien agir dans les quelques états qui apparaissent souvent en début de partie est beaucoup plus intéressant que bien agir dans un état improbable que l'on ne rencontrera peut-être jamais (d'où par exemple le travail sur les ouvertures aux échecs). Pour mesurer l'importance accordée à un état, on définit une fonction de poids homogène à une densité de probabilité[3] et qui donc somme à un : $\rho \in [0,1]^{\mathcal{S}}$ telle que $\sum_{s \in \mathcal{S}} \rho(s) = 1$. La mesure de l'erreur d'une politique de classification π^C se basant uniquement sur la base d'entraînement (avec $\mathbb{1}()$ la fonction indicatrice) :

3. Par abus de notation, nous allons identifier loi de probabilité et densité de probabilité, ce qui nous amènera à écrire des choses comme $s \sim \rho$, même si ρ est défini comme une densité et non comme une loi.

$$\epsilon_C^{emp} = \frac{1}{N} \sum_{(s_i, a_i) \in D_{sa}^{\pi^E}} \mathbb{1}(\pi^C(s_i) \neq a_i) \tag{2.3}$$

est différente de celle que l'on cherche réellement à optimiser :

$$\epsilon_C^{\rho} = \sum_{s \in \mathcal{S}} \rho(s) \mathbb{1}(\pi^C(s) \neq \pi^E(s)) \tag{2.4}$$

$$= \mathbb{E}\left[\mathbb{1}(\pi^C(s) \neq \pi^E(s)) \middle| s \sim \rho \right]. \tag{2.5}$$

Il existe des conditions de consistance du principe de minimisation empirique du risque. Cependant lorsque la taille de la base d'apprentissage et la complexité du modèle ne sont pas compatibles, les problèmes de sur-apprentissage peuvent apparaître.

L'hypothèse de base en discrimination est que la distribution ρ pour laquelle on cherche à minimiser l'erreur de classification est la même que celle selon laquelle on échantillonne les données. Il est alors possible de se baser sur le risque empirique pour résoudre le problème de la classification.

Il peut être nécessaire de choisir une autre distribution ρ sur laquelle optimiser la classification. On peut, comme on l'a vu intuitivement, favoriser les états de départ. Il est possible de tenter d'estimer à partir d'une base d'exemples la vraie distribution des états qui seront soumis au contrôle de l'agent (qui peut être différente de la distribution à laquelle l'expert est confronté). On peut également, pour certains environnements, accorder plus d'importance à certains états critiques où une erreur aurait des conséquences fâcheuses (si l'on dispose de cette connaissance *a priori*).

2.1.3 *Attributs*

La capacité d'une méthode d'imitation à extrapoler un contrôle satisfaisant à partir de données dont la distribution n'est pas forcément celle sur laquelle il nous importe que l'agent soit performant s'appelle la capacité de généralisation. La capacité de généralisation d'un modèle est affectée par la manière dont l'espace d'état lui apparaît. Les descripteurs des données sont souvent choisis de manière à permettre une modélisation linéaire[4] , ceci afin de faciliter leur exploitation. Rien ne garantit qu'une fonction linéaire dans l'espace d'état-action $\mathcal{S} \times \mathcal{A}$ dispose d'un pouvoir de représentation en adéquation avec la complexité des objets à approximer. Pour élargir l'espace d'hypothèses, on va choisir de travailler avec des fonctions linéaires dans un espace d'attributs Φ de dimension d_ϕ qui est l'image de l'espace d'état-action $\mathcal{S} \times \mathcal{A}$ par une fonction vectorielle d'attribut $\phi \in (\mathbb{R}^{d_\phi})^{\mathcal{S}\times\mathcal{A}}$. Par exemple, le classifieur que nous étudierons sous-section 2.1.4 recherche une fonction de score telle que :

4. Comme nous le verrons tout au long du manuscrit, la méthode d'approximation que nous nous préparons à expliquer peut être utilisée pour la fonction de score d'un classifieur, la fonction de valeur ou de qualité en AR et pour la fonction de récompense et l'attribut moyen en ARI.

$$q_\omega = \omega^T \phi_q. \tag{2.6}$$

Les approximations linéaires comme celle-ci ont plusieurs propriétés qui les rendent populaires :

- elles permettent de remplacer la recherche dans un espace fonctionnel par une recherche dans l'espace des paramètres $\mathbb{R}^{d_\phi}$ (elles ne sont bien sûr pas seules à permettre cela) ;
- deux vecteurs de paramètres ω et ω' proches dans cet espace (tels que $||\omega - \omega'||_2$ soit petit) correspondent à deux fonctions q_ω et $q_{\omega'}$ également proches. Par contraste, dans les réseaux de neurones par exemple, la non linéarité des fonctions d'activation peut faire que deux jeux de paramètres semblables correspondent à deux fonctions radicalement différentes ;
- les fonctions linéaires sont aisément dérivables en fonction du vecteur de paramètre. Le gradient est le vecteur d'attribut :

$$\nabla_\omega q_\omega = \nabla_\omega(\omega^T \phi_q) = \phi_q; \tag{2.7}$$

- la mise en œuvre informatique est triviale, il s'agit du produit scalaire du vecteur de paramètres et du vecteur d'attributs.

Si l'on dispose d'une fonction d'attribut $\psi \in (\mathbb{R}^{d_\psi})^{\mathcal{S}}$ sur l'espace d'état, une technique classique pour obtenir une fonction d'attribut sur l'espace d'état-action consiste à distribuer la représentation sur les différentes actions. D'un vecteur de dimension d_ψ, on passe à un vecteur de dimension $d_\phi = |A|\, d_\psi$ (où $|\cdot|$ dénote le cardinal d'un ensemble)

en définissant :

$$\phi(s,a) = \begin{pmatrix} \mathbb{1}(a = a_1)\psi(s) \\ \vdots \\ \mathbb{1}(a = a_{|\mathcal{A}|})\psi(s) \end{pmatrix}. \tag{2.8}$$

Le choix d'une bonne fonction d'attribut sur l'espace d'état est extrêmement dépendant du problème, néanmoins dans de nombreux cas deux techniques simples donnent de bons résultats. Dans le cas d'un espace d'état fini de taille raisonnable, il est possible de définir une fonction d'attribut binaire en associant un unique indice à chaque état. Le vecteur d'attribut d'un état est nul partout sauf en l'indice associé à l'état :

$$\psi(s) = \begin{pmatrix} \mathbb{1}(s = s_1) \\ \vdots \\ \mathbb{1}(s = s_{|\mathcal{S}|}) \end{pmatrix}. \tag{2.9}$$

Un avantage de ce schéma est qu'il permet une représentation exacte de la fonction de score. En effet le produit $q_\omega(s,a) = \omega^T \phi_q(s,a)$ revient à isoler la composante de ω correspondant à l'unique indice associé au couple (s,a). Les deux gros désavantages sont l'incapacité de ce schéma à passer à l'échelle et l'absence totale de structure : on aura beau disposer d'énormément d'information sur les "voisins" d'un élément de l'espace, tant que l'on aura pas vu précisément cet élément dans la base d'exemple, c'est la valeur d'initialisation de la coordonnée correspondante dans ω qui sera utilisée.

Pour les espaces continus, une paramétrisation usuelle consiste à paver l'espace de fonctions à base radiale, telles les gaussiennes. On assigne un nombre g_j à chacune des dimensions $0 < j \leq d_\mathcal{S}$ de l'espace d'état et l'on construit un maillage de $d_\psi = \prod_{j=1}^{d_\mathcal{S}} g_j$ points $\mathfrak{m}_k$, $0 < k \leq d_\psi$ équirépartis dans l'espace qui seront les centres des d_ψ composantes gaussiennes de la fonction d'attribut. L'écart-type pour une dimension j peut être choisi par exemple comme :

$$\sigma^j = \frac{\max(s_j) - \min(s_j)}{2g_j}, \text{ avec } s_j \text{ la } j^{\text{ième}} \text{ composante de } s. \tag{2.10}$$

En notant :

$$\mathfrak{G}_\sigma^\mathfrak{m}(s) = \exp\left(-\sum_{j=1}^{d_\mathcal{S}} \frac{(s_j - \mathfrak{m}_j)^2}{2(\sigma_j)^2}\right), \tag{2.11}$$

on obtient finalement la fonction d'attribut suivante après l'ajout d'une composante constante [5] :

$$\psi(s) = \begin{pmatrix} \mathfrak{G}_\sigma^{\mathfrak{m}_1}(s) \\ \vdots \\ \mathfrak{G}_\sigma^{\mathfrak{m}_{d_\psi}}(s) \\ 1 \end{pmatrix}. \tag{2.12}$$

5. Formellement, on utilise du fait de l'ajout de ce terme constant une modélisation affine plutôt que linéaire.

Contrairement à la fonction d'attribut binaire précédente, celle-ci possède une structure spatiale. Les scores de deux états topologiquement proches subiront l'influence de la même composante du vecteur de paramètres. Le nombre de gaussiennes du réseau croît de manière exponentielle avec le nombre de dimensions. Une illustration de ce type de vecteur d'attributs sur l'espace d'état bidimensionnel du *mountain-car* [6] est présentée Figure 2.1.

6. Il s'agit d'un problème jouet où une voiture doit sortir d'une vallée, mais n'est pas assez puissante pour gravir la côte du premier coup. Ce problème est décrit en section B.2.

Lorsque cela s'avère nécessaire, l'opérateur peut effectuer un travail d'ingénierie en construisant un vecteur d'attribut spécifique au problème courant, qui tient compte des informations *a priori* disponibles sur celui-ci. Par exemple pour la fonction de valeur

d'un agent sur le problème du *mountain-car*, on pourrait envisager de calculer l'énergie cinétique et l'énergie potentielle. Ce n'est cependant pas la route que nous allons suivre. Notre but étant d'établir des modèles génériques, nous utiliserons dans nos expériences les attributs gaussiens et binaires que nous venons de décrire.

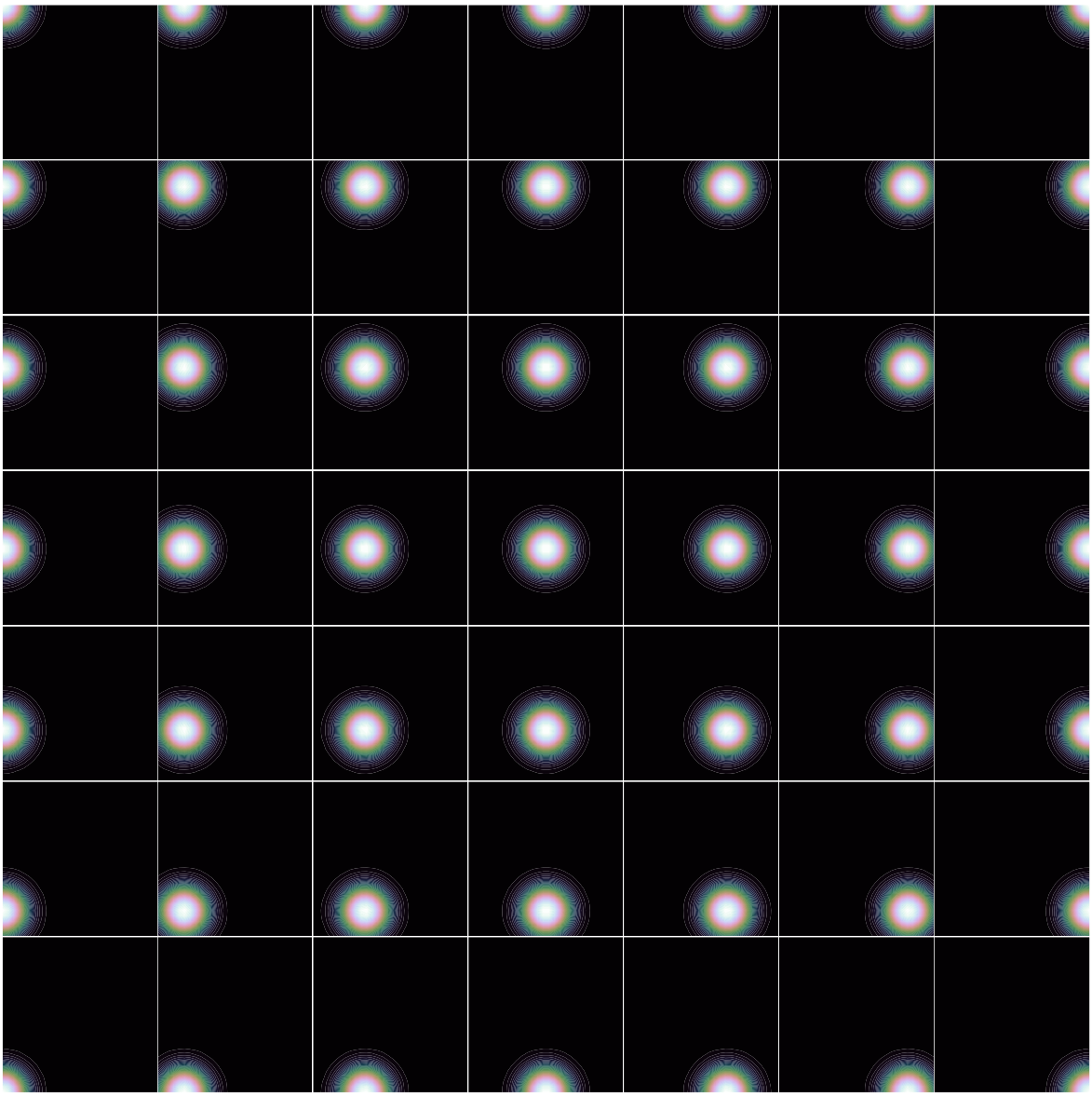


FIGURE 2.1: Attributs gaussiens sur le problème du *mountain-car*. L'espace d'état est pavé de $7 \times 7 = 49$ gaussiennes elliptiques dont les centres sont répartis à équidistance. Les variances sont les mêmes pour toutes les gaussiennes et dépendent de la plage de valeurs sur une dimension. Toute fonction aux variations raisonnables peut être approximée de manière correcte par une somme pondérée de ces gaussiennes. Cette représentation est à comparer à la Figure 2.2 où est présenté l'attribut moyen qui, lui, prend en compte la dynamique temporelle du problème.

2.1.4 *Classification à marge structurée*

Illustrons ce propos par l'étude d'une approche de classification qui utilise une fonction de score linéairement paramétrée sur l'espace d'attribut état-action Φ [Tas+05].

Le principe quasi-ubiquitaire en classification[7] de la fonction de score est le suivant : à chaque couple état-action une fonction q associe un score. Pour associer une action à un état, le classifieur passe simplement en revue toutes les actions (on voit donc l'intérêt d'un petit espace d'action) et choisit celle qui associée à cet état obtient le score le plus haut :

7. Les arbres de décision formant un contre-exemple notable [SL91].

$$q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}, \tag{2.13}$$

$$\forall s, \pi^C(s) \in \arg\max_{a\in\mathcal{A}} q(s,a). \tag{2.14}$$

Apprendre une bonne fonction de score permet donc de résoudre le problème de classification. L'approche proposée dans [Tas+05] prend le parti d'une fonction de score linéaire :

$$q_\omega(s,a) = \omega^T \phi_q(s,a). \tag{2.15}$$

Ce que nous cherchons maintenant est donc un bon vecteur de paramètres ω.

Considérant que nous disposons d'un vecteur d'attributs permettant de continuer, examinons à des fins d'illustration le problème que TASKAR, CHATALBASHEV, KOLLER et GUESTRIN [Tas+05] se proposent de résoudre, problème que nous retrouverons chapitre 5. Il s'agit d'un problème d'optimisation convexe sous contraintes[8] :

8. Pour voir que les variables ζ_i sont positives, on peut poser $a = a_i$ (et considérer que $\mathfrak{l}(s_i, a_i) = 0$).

$$\min_{\zeta\in\mathbb{R}_+} \frac{\lambda}{2}\|\omega\|^2 + \frac{1}{N}\sum_{i=1}^{N}\zeta_i \tag{2.16}$$

$$\text{sous les contraintes } \forall i, q_\omega(s_i,a_i) \geq \max_{a\in\mathcal{A}}(q_\omega(s_i,a) + \mathfrak{l}(s_i,a)) - \zeta_i. \tag{2.17}$$

La fonction objectif cherche naturellement à réduire les variables d'écart ζ_i tandis que les contraintes sont telles que la valeur prise par la fonction de score q_ω en un couple (s_i, a_i) correspondant à une décision experte est supérieure ou égale au meilleur score. Elle doit être non seulement supérieure au score seul, mais supérieure d'une certaine marge $\mathfrak{l}(s_i, a)$ qui contribue à donner à ce classifieur sa capacité de généralisation. En effet, on constate que si $\mathfrak{l}$ est uniformément nulle, alors parvenir à minimiser parfaitement la fonction de coût se réduit à apprendre par cœur la base d'exemples, c'est-à-dire à probablement subir les effets du sur-apprentissage. Fixer

$$\mathfrak{l}(s_i,a) = \begin{cases} 0 & \text{si } a = a_i \\ 1 & \text{si } a \neq a_i \end{cases} \tag{2.18}$$

permet de donner aux choix de l'expert un score strictement supérieur aux scores des autres choix. TASKAR, CHATALBASHEV, KOLLER et GUESTRIN [Tas+05] précisent qu'il est possible d'adapter la marge $\mathfrak{l}$ en fonction de la qualité des choix alternatifs, un bon choix correspondant à une petite marge. Nous verrons qu'en pratique la marge binaire que nous venons de suggérer fonctionne assez bien pour les problèmes qui nous intéressent.

La résolution informatique d'un problème d'optimisation convexe sous contraintes pouvant être malaisée, nous en cherchons une formulation plus simple (qui revient à celle employée dans [RBS07]) en constatant que si une contrainte de bon classement est violée, alors compte-tenu de la nature de la contribution empirique à la fonction

objectif, la valeur de la variable d'écart correspondante est obtenue en traitant ladite contrainte comme une contrainte-égalité :

$$\zeta_i = \max_{a \in \mathcal{A}} (q_\omega(s_i, a) + \mathfrak{l}(s_i, a)) - q_\omega(s_i, a_i). \tag{2.19}$$

En posant :

$$a_i^* = \arg\max_{a \in \mathcal{A}} (q_\omega(s_i, a) + \mathfrak{l}(s_i, a)) \tag{2.20}$$

$$= \arg\max_{a \in \mathcal{A}} (\omega^T \phi_q(s_i, a) + \mathfrak{l}(s_i, a)), \tag{2.21}$$

et en faisant monter les contraintes dans la fonction objectif, on obtient alors une simple fonction de coût à minimiser :

$$J(\omega) = \frac{1}{N} \sum_{i=1}^{N} (q_\omega(s_i, a_i^*) + \mathfrak{l}(s_i, a_i^*) - q_\omega(s_i, a_i)) + \frac{\lambda}{2} \|\omega\|^2 \tag{2.22}$$

$$= \frac{1}{N} \sum_{i=1}^{N} (\omega^T \phi_q(s_i, a_i^*) + \mathfrak{l}(s_i, a_i^*) - \omega^T \phi_q(s_i, a_i)) + \frac{\lambda}{2} \|\omega\|^2. \tag{2.23}$$

Cette fonction n'est pas différentiable à cause de l'opérateur non-linéaire max caché dans le terme a_i^*, mais la généralisation du gradient qu'est le sous-gradient permet de contourner cette difficulté. Le sous-gradient de la fonction de coût est :

$$\nabla J(\omega) = \sum_{i=1}^{N} (\phi_q(s_i, a_i^*) - \phi_q(s_i, a_i)) + \lambda \omega, \tag{2.24}$$

il est donc possible de résoudre le problème d'optimisation original en effectuant une simple descente de sous-gradient pour minimiser la fonction de coût $J(\omega)$, comme cela est décrit dans l'algorithme 1.

Algorithme 1: Algorithme de classification structurée

Entrées : Une base d'entraînement établie par l'expert $D_{sa}^{\pi^E}$;
Données : Une fonction d'attribut ϕ_q;
Sorties : Une règle de décision π^C

1 Initialiser ω arbitrairement;
2 Procéder à la descente de sous-gradient dont le sous-gradient est :

$$\nabla J(\omega) = \sum_{i=1}^{N} \left[\phi_q \left(s_i, \max_{a \in \mathcal{A}} \omega^T \phi(s_i, a) + \mathfrak{l}(s_i, a) \right) - \phi_q(s_i, a_i) \right] + \lambda \omega$$

retourner

$$\pi^C : s \to \pi^C(s) \in \arg\max_{a \in \mathcal{A}} \omega^T \phi_q(s, a)$$

Nous avons présenté cette technique de classification plus en détail à des fins d'illustration de l'importance du choix des attributs, et également car nous la retrouverons sous-section 5.2.2 lorsque nous nous intéresserons à une nouvelle technique d'ARI.

Il existe bien d'autres moyens de faire de la classification, comme par exemple les SVM.

2.1.5 SVM

Une SVM est un séparateur linéaire. Plus précisément, une SVM va chercher, dans un espace de représentation des entrées, l'hyperplan de marge maximale séparant les

données : d'un côté de l'hyperplan se trouvent toutes les entrées associées à une certaine étiquette, et de l'autre côté se trouvent les entrées associées à l'autre étiquette. Il faut positionner l'hyperplan dans l'espace de représentation de manière à maximiser la distance entre celui-ci et les données.

Il est illusoire de croire que l'espace de description est construit d'une manière qui permette à un hyperplan de séparer les exemples sans erreur. Le classifieur à fonction de score paramétrée linéairement que nous avons décrit précédemment compense cela par une projection dans un espace d'attributs (ou espace de représentation) Φ. Les SVM utilisent une technique similaire, l'idée est de remplacer les produits scalaires $\langle s, s' \rangle_{\mathcal{S}}$ qui apparaitraient dans l'algorithme par un produit scalaire $\langle \psi(s), \psi(s') \rangle_{\Psi}$ dans l'espace d'attributs Ψ, dans l'espoir que les données soient linéairement séparables dans cet espace. L'astuce, dite astuce du noyau [ABR64 ; BGV92] est d'observer que les données n'apparaissent dans l'algorithme que par le biais de leur produit scalaire ; plutôt que de définir explicitement cet attribut ψ et de calculer le produit scalaire associé, un noyau κ est défini. Pour peu que κ satisfasse certaines conditions [9], alors il existe [Mero9 ; BT04] une fonction ψ telle que [10] :

$$\kappa(s, s') = \langle \psi(s), \psi(s') \rangle_{\Psi} . \tag{2.25}$$

9. À savoir être continu, symétrique et semi-défini positif.

10. Le noyau κ engendre un *Reproducing Kernel Hilbert Space* (RKHS) [BT04].

Un noyau populaire, le noyau gaussien :

$$\kappa(s, s') = \exp\left(-\frac{\|s - s'\|^2}{2\sigma^2} \right), \tag{2.26}$$

permet de manipuler le produit scalaire $\langle \psi(s), \psi(s') \rangle_{\Psi}$ où l'espace d'attributs Ψ engendré par ψ est de dimension infinie.

Le modèle que nous venons de décrire est celui de la SVM à marge dure [BGV92]. Comme dans le cas du classifieur à marge de la sous-section 2.1.4, il est possible d'introduire des variables d'écart. On passe alors au modèle de la SVM à marge douce [CV95].

Pour l'imitation, la multiplicité des actions possibles appelle l'utilisation d'une SVM multi-classe (M-SVM). Un modèle unifiant les machines de ce type est proposé dans [Gue12]. Pour simplifier l'exposé nous nous sommes cantonné au cas bi-classe. Le classifieur à marge de la sous-section précédente peut être instancié (si l'on choisit convenablement les valeurs des hyper-paramètres) comme un cas particulier d'une SVM, mais les deux modèles ne sont pas équivalents dans leur forme générique.

Les SVM sont très populaires et de nombreux logiciels les mettent en œuvre. Il est donc facile de les mettre en œuvre sur les problèmes jouet que nous allons traiter ; Ce sont des classifieurs pour lesquels la sélection de modèle est assez bien maîtrisée (BONIDAL [Bon13] a récemment présenté une thèse abordant ce sujet).

2.1.6 *Imitation par apprentissage supervisé de la politique*

Apprendre la politique de l'expert de manière supervisée à l'aide d'une base d'exemples peut s'avérer efficace, comme le démontrent plusieurs approches. Dans [RBS07], les auteurs utilisent le classifieur à marge décrit plus haut pour apprendre une politique experte pour un problème de locomotion quadrupède et sur un problème de manipulation d'objets. Le choix des attributs est automatisé grâce à une technique de *boosting* similaire à [Frio1 ; Mas+99].

Le boosting permet de déplacer de manière intelligente le problème du choix des attributs, sans le régler totalement. Il reste en effet à construire l'espace où choisir les

nouveaux attributs. Un espace trop simple ne permettrait pas de minimiser efficacement la fonction de coût, tandis qu'un espace trop riche permettrait de l'annuler, mais sans doute au prix d'un sur-apprentissage aux conséquences fâcheuses. C'est donc cet espace qui doit être construit afin de donner au classifieur ses capacités de généralisation. RATLIFF, BAGNELL et SRINIVASA [RBS07] proposent d'utiliser un réseau de neurones.

Plus brutale [11], l'approche de LECUN, MULLER, BEN, COSATTO et FLEPP [LeC+06] utilise un *Convolutional Neural Network* (CNN) (réseau de convolution) à 6 couches pour apprendre une association directe entre une image (stéréo) et un angle de braquage (la tâche à apprendre est la conduite d'un véhicule en terrain libre). Le problème de la généralisation est résolu en exigeant une base d'entraînement couvrant au maximum l'espace d'état. Les auteurs ne cachent pas la difficulté de constituer une telle base qui doit réunir des conditions de terrain et d'illumination variées tout en exigeant un comportement extrêmement cohérent et prédictible de la part de l'opérateur humain et ce sur un grand nombre de trajectoires (il faut réunir près d'une centaine de milliers d'échantillons). En contrepartie de ces efforts, la technique proposée est robuste et ne nécessite aucun travail d'ingénierie au niveau des attributs, puisque la politique apprise associe directement la sortie du capteur à la consigne de l'actuateur du robot. Bien que cela soit moins problématique aujourd'hui avec l'augmentation de puissance des équipements embarqués, elle semble également plus rapide (dans l'exploitation, non dans l'apprentissage) que l'état de l'art de l'époque. Elle améliore les résultats notamment par rapport à *Autonomous Land Vehicle In a Neural Network* (ALVINN) [Pom93] en ceci que la résolution des caméras peut être augmentée sans une trop grande explosion du nombre de connexions du réseau grâce à l'usage de la convolution et non d'un réseau complètement connecté, et que la tâche apprise est plus difficile, il s'agit de conduire en terrain libre et non de suivre une route.

11. Puisqu'elle repose sur l'utilisation d'une base d'exemple exhaustive.

Nous venons de voir deux outils différents (*boosting* et réseau de convolution) permettant d'apprendre une politique sans contrôler le système, de manière supervisée, avec une intervention humaine minimale : soit l'on dispose de suffisamment de données pour que le risque empirique soit proche du risque réel, soit l'on construit des attributs tels que l'apprentissage au mieux (en minimisant une fonction de coût exprimée sur les données) ne soit pas un apprentissage par cœur, mais un apprentissage généralisant sur tout l'espace d'état. Apprendre une politique de manière locale, c'est-à-dire en se concentrant trop sur une base de données ne couvrant pas tout l'espace d'état, n'est pas satisfaisant. Cela donne un résultat fragile, l'agent sera en effet pris au dépourvu s'il a à contrôler le système dans une configuration différente de celles sur lesquelles il a été entraîné : il ne dispose ni d'information relative au comportement de l'expert dans une telle situation, ni d'information sur la tâche à accomplir qui lui permettraient de déduire ce que pourrait être ce comportement.

L'idée de demander ces échantillons de manière interactive a été proposée afin de minimiser la quantité de données nécessaire à l'apprentissage de la politique experte. Un exemple d'une telle approche est décrit par CHERNOVA et VELOSO [CV07]. Des *Gaussian Mixture Model* (Modèle à base de Mélange de Gaussiennes) (GMM) sont appris à partir d'une base de données experte de départ, puis l'agent applique la politique apprise tout en demandant à l'expert, lorsque l'incertitude est trop grande, de lui fournir un échantillon supplémentaire. Cette approche permet de limiter la redondance de la base d'entraînement et de guider l'échantillonnage vers les zones intéressantes de l'espace d'état, ce qui est également une solution au problème de la généralisation : quand l'agent ne sait pas généraliser, il demande à l'expert. Cela est connu sous le nom d'apprentissage actif [CGJ96].

Une autre approche d'apprentissage actif est l'algorithme Stochastic Mixing Iterative Learning (SMILe) proposé par Ross et Bagnell [RB10]. A chaque itération t une politique mixante π_t [12] est définie à l'aide d'une politique de classification π^C apprise sur une base de données experte. Cette base de données est construite sur la distribution d'états induite par l'application de la politique mixante de l'itération précédente, π_{t-1}. La politique mixante à l'itération t est une pondération de trois politiques : la politique mixante de l'itération précédente π_{t-1}, la politique apprise par le classifieur à l'itération courante π^C et la politique initiale π_0 qui se réfère systématiquement à l'expert. L'algorithme Dataset Aggregation (DAGGER) (Ross, Gordon et Bagnell [RGB10]) maintient une base d'entraînement de plus en plus grande grâce à une politique mixante entre une politique de classification et la politique de l'expert. Les états-actions traversés par la politique mixante courante ayant donné lieu à une décision de l'expert sont ajoutés à la base d'entraînement et à l'itération suivante le classifieur est entraîné sur la nouvelle base. Le poids accordé à la politique de l'expert diminue pour laisser petit à petit le contrôle à l'agent. La logique derrière ces deux approches est d'éviter de faire de l'apprentissage supervisé sur la distribution fixe de l'expert ρ_E, pour la remplacer par la distribution que l'agent rencontrera vraiment, et qui tient compte des erreurs qu'il peut parfois commettre. Grâce au schéma interactif où l'expert donne peu à peu le contrôle à l'agent, l'expert est parfois entraîné loin de sa distribution et peut alors montrer à l'agent comment corriger l'erreur que celui-ci a commise.

12. Une politique mixante est une sorte de méta-politique qui choisit aléatoirement quelle politique exécuter parmi un ensemble de politiques, généralement de manière non uniforme, certaines politiques étant préférées à d'autres.

L'apprentissage direct de la politique experte est parfois intégré à un cadre plus large, où les notions de hiérarchie et de but apparaissent.

La classification par arbre de décision a été appliquée à l'apprentissage d'un plan de vol par Sammut, Hurst, Kedzier et Michie [Sam+92]. L'application est ambitieuse ; *apprendre* [13] à piloter un avion, même en simulation, n'est pas une mince affaire puisqu'il faut en temps réel prendre en compte un grand nombre de facteurs pour décider d'une action parmi un éventail assez large. L'apprentissage automatique nécessite un grand nombre d'échantillons. Un comportement cohérent est exigé de l'expert humain (à un point tel que les démonstrations de deux experts ne peuvent être mélangées en une seule base d'entraînement). L'aspect automatique de l'approche est limité à l'apprentissage d'une politique par phase de vol. La détection de la phase de vol courante et donc le choix de la politique de contrôle à appliquer est effectué par des règles d'origine humaine.

13. La conception par des humains d'un système de pilotage automatique précède l'ordinateur puisque la démonstration d'un prototype a été faite à Paris en 1914.

De fait, cette approche a été le point de départ de nombreuses améliorations. Le travail présenté par Stirling [Sti95] (appelé *Compressed Heuristic Universal Reaction Planners* (CHURPs)) consiste à déduire un contrôleur à partir d'une description du modèle d'évolution du système et du but à atteindre. Pour automatiser la création de ces descriptions, tâche réclamant un travail difficile car nécessitant de décrire des mécanismes précis à l'aide d'un langage contraignant, Bain et Sommut [BS00] proposent d'utiliser les données de l'expert. Les règles complexes ainsi apprises étant ajoutées à l'espace d'action, il est possible d'apprendre de manière automatique un classifieur plus concis que celui de Sammut, Hurst, Kedzier et Michie [Sam+92], et nécessitant moins de données expertes. L'architecture proposée utilise la logique du premier ordre et donc le raisonnement symbolique. Cela permet d'introduire explicitement des connaissances expertes dans le système. Ces connaissances peuvent être acquises semi-automatiquement : les prédicats sont construits à la main et les paramètres sont appris grâce aux données de vol comme le proposent Srinivasan et Camacho [SC98]. La sémantique des symboles (ici, virage, altitude, trajectoire de vol, etc.) est très liée au problème concerné. Retrouver la puissance des techniques d'apprentissage symbolique

sur un autre problème nécessite d'effectuer de nouveau le difficile travail de définition des symboles et prédicats. D'autres éléments potentiellement rédhibitoires sont la mauvaise gestion des fluctuations aléatoires dues à la dynamique et la difficulté d'exprimer la tâche à accomplir en utilisant un langage symbolique. Dans une approche hybride symbolique/numérique, SHIRAZ et SAMMUT [SS97] proposent à l'expert soit de décrire la tâche symboliquement, soit d'en démontrer l'exécution. Les phases les plus délicates (par exemple l'atterrissage) n'ont pu être décrites et ont été démontrées. La facilité d'exploitation des règles symboliques rentre en conflit avec la difficulté qu'il y a à les définir, à l'inverse la relative facilité de génération d'une base d'exemples se heurte à la difficulté qu'il y a à généraliser à partir de celle-ci.

Une autre approche utilise les notions de hiérarchie et de but, mais de manière quelque peu différente. Plutôt que d'utiliser la logique des prédicats, ce sont les principes de programmation impérative qui se voient assistés par l'apprentissage supervisé. Dans [SND06], ce sont les *k-plus proches voisins* (en anglais *k-Nearest Neighboors*) (*k*-NN) qui sont utilisés pour l'apprentissage supervisé d'une politique. Les attributs sont construits à la main à partir des valeurs de sortie des capteurs du robot, et portent une sémantique forte et explicite (distance, angle), donc pratique pour l'exploitation par un opérateur humain. Les problèmes de généralisation de l'apprentissage supervisé sont contournés par l'intégration dans un cadre beaucoup plus riche : l'opérateur humain peut élargir l'espace d'action à volonté, soit en définissant une séquence d'actions qui seront exécutées en série de manière déterministe, soit en proposant des exemples du comportement souhaité en précisant ou non un état-but correspondant à la situation dans laquelle on souhaite voir le robot une fois la politique exécutée. Ces exemples servent alors à l'apprentissage d'une politique de manière supervisée, cette politique est ajoutée en tant qu'action et son exécution pourra être déclenchée dans le cadre d'une autre politique, de niveau d'abstraction plus grand. Cette hiérarchisation des comportements permet de limiter l'effort humain, d'optimiser l'utilisation des exemples et de rapidement mettre en place des comportements complexes par la création de nouveaux niveaux d'abstraction.

L'apprentissage supervisé est dans les approches que nous venons de citer utilisé comme sous routine d'un système beaucoup plus large, dans lequel l'expertise humaine explicite reste le moyen central permettant la généralisation des comportements.

Le principal problème de l'apprentissage direct de la politique de manière supervisé est, pour reprendre le terme de [RSB09], sa *myopie*. Pour compenser le fait que l'on travaille au niveau d'abstraction le plus bas, celui du choix immédiat d'une action en fonction des informations contenues dans un état transitoire, les approches que nous venons de détailler font apparaître en filigrane la notion de but : l'expert n'agit en effet pas à tâtons mais dirige le système en fonction de critères qu'il paraît difficile d'exprimer au niveau d'une simple politique. On se repose donc sur une formulation plus ou moins explicite (dans le choix des attributs, dans la définition de la base d'exemples, dans l'introduction de règles logiques ou dans la définition d'une hiérarchie) de ce but, mais toujours d'origine humaine. Nous allons voir qu'il est possible de formaliser cette notion de but tout en continuant à travailler avec une politique et des échantillons semblables à ceux auxquels nous sommes habitués. Nous verrons par la suite que le but de l'expert, formalisé de cette manière, peut alors être automatiquement déduit d'une base d'échantillons acquise une fois pour toutes.

2.2 *Cadre des PDM pour la prise de décisions séquentielles*

Pour comprendre ce but de l'expert qu'il nous importe de connaître, ce n'est pas au niveau du choix état-action que décrit la politique qu'il faut regarder, mais à un niveau d'abstraction plus élevé : la dynamique que la politique de l'expert impose au système. La notion qui nous manque pour entamer le raisonnement est celle de l'effet d'une action. Nous ne nous sommes préoccupés que du choix de l'action en fonction de l'état courant sans nous soucier de ce que ce choix allait imposer comme contraintes sur le prochain état que l'agent va rencontrer. Afin de pouvoir considérer la politique de l'expert non plus comme un ensemble décousu d'associations état-action, mais comme un outil capable de produire des séquences d'actions pertinentes au sens d'un critère long terme, nous formalisons la notion de dynamique temporelle.

2.2.1 *Dynamique temporelle*

L'agent (qu'il s'agisse de l'expert ou d'un agent artificiel que l'on entraîne) aux manettes du système contrôle celui-ci non pas ponctuellement de temps à autres mais de manière cohérente sur un laps de temps durant lequel il devra opérer des actions de contrôle les unes après les autres. Il est donc naturel d'indexer ces actions et les états traversés par un indice temporel $t \in [\![0; \infty]\!]$. Par exemple, cette formulation n'impose pas de pas d'échantillonnage constant, il s'agit ici d'ordonner les états et actions par ordre de causalité, ce qui incidemment correspond [14] à un indice temporel croissant, non pas de transcrire avec quelque fidélité les problèmes de l'échantillonnage temporel. Qui plus est cette formulation correspond à la réalité du contrôle numérique, intrinsèquement discret.

14. À moins que *Doctor Who* et *Retour vers le futur* ne soient des documentaires.

Pour prendre en compte les imperfections de la modélisation ou plus simplement parfois la nature réellement stochastique du problème, les effets d'une action qui sont décrits par une loi de probabilité, qui, informée d'un état s_t et d'une action a_t, prédit vers quel état s_{t+1} le système va transiter. On note cela (encore une fois en identifiant loi de probabilité et densité de probabilité) [15] :

$$s_{t+1} \sim p\left(\cdot | s_t, a_t\right), p \in [0;1]^{\mathcal{S}\times\mathcal{A}\times\mathcal{S}}. \tag{2.27}$$

15. Avec $f \in B \times C \times D^A$, la notation $f(b, \cdot, d)$ désigne la fonction de C dans $A : c \to f(b, c, d)$.

On constate que l'information sur l'état vers lequel on transite ne dépend que de l'état courant et de l'action courante, et non de la trajectoire. C'est la propriété de Markov qui donne son nom au PDM. La répétition du cycle consistant à choisir une action puis à transiter vers un nouvel état où l'agent choisit une action qui le fera transiter vers un nouvel état (etc.) forme une trajectoire. Les probabilités de transitions contraintes par une politique π peuvent être dans le cas fini $|\mathcal{S}| < \infty$ représentées [16] par une matrice stochastique de taille $|\mathcal{S}| \times |\mathcal{S}|$:

16. En utilisant l'isomorphisme naturel entre $\mathcal{S}$ et $[\![1, |\mathcal{S}|]\!]$.

$$P^\pi = \left(p\left(s' | s, \pi(s)\right)\right)_{s,s'}, \tag{2.28}$$

où l'on note : $(f(i,j))_{i,j}$ la matrice dont le terme à la ligne i et à la colonne j est $f(i,j)$. Un agent de politique π va visiter certains états plus que d'autres. Pour quantifier cela, il est possible d'utiliser la matrice des probabilités de transition que nous venons juste de définir. Le terme à la ligne s et à la colonne s' est la probabilité $p\left(s' | s, \pi(s)\right)$ que l'agent se trouve dans l'état s' au temps $t+1$ s'il était en s au temps t. Si l'on multiplie la matrice P^π par elle même, le terme général (ligne s, colonne s'') du résultat est :

$$\sum_{s' \in \mathcal{S}} p\left(s' | s, \pi(s)\right) p\left(s'' | s', \pi(s')\right), \tag{2.29}$$

il s'agit de la probabilité pour l'agent de passer de s à s'' en deux pas de temps. Si (et seulement si) l'agent peut espérer se trouver en chacun des états en temps fini, alors ce que l'on appelle la distribution stationnaire ρ_π existe et est unique [Nor98]. Elle est définie telle que :

$$\rho_\pi{}^T P^\pi = \rho_\pi{}^T. \tag{2.30}$$

On peut donc la voir comme la probabilité pour l'agent de se trouver en un certain état, après un temps infini, quel que soit l'état de départ. Cette condition d'ergodicité que nous plaçons généralement sur le PDM contraint par la politique de l'expert implique l'existence de la distribution stationnaire de l'expert ρ_E. Un PDM contraint par une politique correspond à une chaine de Markov.

2.2.2 *Récompense et valeur*

Dans les approches vues précédemment, le but était défini comme des valeurs spécifiques que doivent prendre certaines composantes de l'état (par exemple pour le pilotage, une certaine altitude) ou certaines valeurs de commandes acceptables dans certaines zones de l'espace d'état. Il est au premier abord assez naturel de définir une consigne comme cela. Pour peu que l'espace d'état-action soit construit d'une manière qui permet l'analyse sémantique, l'opérateur humain n'a pas trop de mal à exprimer ce qu'il souhaite que la machine fasse en définissant les états et actions désirables et ceux qu'il faut éviter. Charge à la machine de trouver comment se placer dans les états désirables en évitant les états problématiques. Nous formalisons cela sous la forme d'une fonction de récompense. Il s'agit d'un jugement local de l'intérêt qu'il y a à se trouver en un certain état ou appliquer certaines actions en certains états :

$$R \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}. \tag{2.31}$$

Nous reviendrons section 3.1 sur la définition de la fonction de récompense.

Il faut maintenant que ce critère local donne lieu à un comportement globalement intéressant. Comment, à l'échelle d'une politique choisissant une action pour un état, parvenir à un contrôle tenant compte de la dynamique complète du système ? Il faut qu'une politique π soit jugée dans son ensemble sur la trajectoire qu'elle impose à l'agent. Mathématiquement nous souhaitons optimiser la valeur de la politique [17] :

$$V_R^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\middle| s_0 = s\right] \tag{2.32}$$

$$\text{avec } \forall t \in [\![1;\infty]\!], s_t \sim p\left(\cdot|s_{t-1},\pi(s_{t-1})\right). \tag{2.33}$$

17. Pour simplifier, à l'avenir nous noterons le conditionnement par la dynamique induite par une politique comme cela : $\mathbb{E}\left[\cdot\,|\,s_0 = s,\pi\right]$

Comme l'horizon temporel est infini, pour s'assurer de la convergence de la somme, le facteur d'amortissement $\gamma \in [0;1[$ est introduit. Une conséquence est la diminution d'attrait des récompenses loin dans le futur au profit de récompenses accessibles plus rapidement. Cela permet de récompenser l'agent qui effectue rapidement sa tâche.

La fonction de valeur, et un peu plus loin la fonction de qualité (Équation 2.43) sont paramétrées par la fonction de récompense R car par la suite nous serons amenés à étudier la valeur selon une certaine récompense d'une politique qui est optimale pour une autre récompense.

L'ensemble de l'espace d'état $\mathcal{S}$, de l'espace d'action $\mathcal{A}$, des probabilités de transitions p, de la fonction de récompense R et du facteur d'amortissement γ forment un PDM $\mathcal{M}$ [Put94]

$$\mathcal{M} = \{\mathcal{S},\mathcal{A},p,R,\gamma\} \tag{2.34}$$

dans lequel le problème de la prise de décision séquentielle pour le contrôle optimal peut être formulé.

Nous recherchons une politique optimale π_R^* telle qu'en tout état sa valeur soit supérieure ou égale à celle de tout autre politique π :

$$\forall s, V_R^{\pi_R^*}(s) \geq V_R^{\pi}(s). \tag{2.35}$$

Pour résoudre ce problème, intéressons nous de plus près à l'expression de la valeur d'une politique, dont la définition, donnée Équation 2.32, peut être transformée en une expression récursive (grâce à la propriété de Markov) :

$$V_R^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p\left(s'|s, \pi(s)\right) V_R^{\pi}(s'). \tag{2.36}$$

C'est l'équation d'évaluation de BELLMAN [Belo3] qui est à l'origine de l'opérateur d'évaluation de Bellman :

$$\text{En définissant : } R_\pi(s) = R(s, \pi(s)) \tag{2.37}$$

$$B_R^{\pi} \in \left(\mathbb{R}^{\mathcal{S}}\right)^{\mathbb{R}^{\mathcal{S}}} \tag{2.38}$$

$$\forall V \in \mathbb{R}^{\mathcal{S}}, B_R^{\pi} V = R_\pi + \gamma P^{\pi} V. \tag{2.39}$$

Cette opérateur est contractant, par conséquence il possède un unique point fixe. L'équation de définition de ce point fixe :

$$V = B_R^{\pi} V \tag{2.40}$$

est exactement la même que l'équation d'évaluation de Bellman (Équation 2.36). L'unique point fixe de l'opérateur B_R^{π} est donc la fonction de valeur de la politique : V_R^{π}. Lorsque l'on dispose des probabilités de transition il est possible de calculer la valeur d'une politique en inversant l'équation de Bellman :

$$V_R^{\pi} = (I - \gamma P^{\pi})^{-1} R_\pi \tag{2.41}$$

Dans l'équation d'évaluation de Bellman, l'action qui fait passer de s à s' est explicitement donnée comme étant $\pi(s)$. Les actions suivantes sont également choisies par la politique π comme l'indique le terme V_R^{π}. Imaginons maintenant que connaissant la valeur d'une politique π, nous soyons chargé pour l'état s de choisir la meilleur action a, qui peut être différente de $\pi(s)$, mais qu'ensuite la politique π reprenne le contrôle. C'est le degré de liberté décrit par la fonction de qualité Q_R^{π} :

$$Q_R^{\pi} \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}} \tag{2.42}$$

$$Q_R^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} p\left(s'|s, a\right) V_R^{\pi}(s'). \tag{2.43}$$

Notre meilleur choix consisterait à maximiser la fonction de qualité, c'est-à-dire à rendre le contrôle à π dans l'état s' dans lequel sa valeur est maximale. En effectuant ce choix sur chacun des états de $\mathcal{S}$, on définit une politique gloutonne :

$$g\left(\pi\right) : s \to \arg\max_{a \in \mathcal{A}} Q_R^{\pi}(s, a). \tag{2.44}$$

Cette agglomération de choix localement optimisés permet un optimisation beaucoup plus générale. La politique gloutonne que nous venons de définir est le meilleur choix pour un problème d'optimisation plus large :

$$g\left(\pi\right) = \arg\max_{\pi'} B_R^{\pi'} V_R^{\pi}. \tag{2.45}$$

L'opérateur associé :

$$B_R^* V = \max_\pi B_R^\pi V \tag{2.46}$$

est l'opérateur d'optimalité de Bellman. Contractant lui aussi, il admet donc un unique point fixe qui se trouve être la fonction de valeur optimale $V_R^{\pi_R^*}$. Toute politique gloutonne vis-à-vis de la politique optimale associée est également une politique optimale. Ainsi :

$$\pi_R^* \in g(\pi_R^*) \tag{2.47}$$

$$\forall s \in \mathcal{S}, \pi_R^*(s) \in \arg\max_{a \in \mathcal{A}} Q_R^{\pi_R^*}(s,a). \tag{2.48}$$

Il est intéressant de noter que, grâce à la fonction de valeur, l'optimisation "myope" répétée à l'échelle du choix d'une action pour un état mène à une optimisation à l'échelle de l'espace d'état complet, au niveau de la politique. Grâce à la prise en compte des probabilités de transitions, la fonction de valeur fait le lien entre le court et le long terme.

Lorsque les probabilités de transitions sont connues sur un espace d'état fini, on peut de manière exacte résoudre le problème du contrôle optimal grâce à la PD [Put94]. Les choses se corsent lorsque ces probabilités de transition sont inconnues ou que les algorithmes de PD deviennent non tractables (espace d'état trop grand, principalement). Il faut alors avoir recours à l'AR [SB98]. L'AR permet, par interactions répétées avec le système, d'apprendre à contrôler celui-ci. Les probabilités de transition étant souvent difficiles à exprimer, nous allons porter notre attention sur l'AR de préférence à la PD.

2.2.3 *Algorithmes d'AR*

La PD supposant les probabilités de transition connues, si l'on connaît la valeur d'une politique, il est possible de calculer la fonction de qualité associée grâce à l'Équation 2.43, et donc d'en déduite un contrôle glouton grâce à l'Équation 2.44.

Puisqu'en AR la dynamique ne peut être connue que par interaction avec le système, on apprend directement la fonction de qualité Q afin de pouvoir en déduire un contrôle glouton sans avoir à passer par l'Équation 2.43.

Il existe plusieurs schémas d'AR. Certains comme par exemple Q-learning [Wat89] adaptent le principe d'itération de la valeur [Belo3]. Une amélioration permettant le fonctionnement hors ligne, à partir d'une base de données acquise une fois pour toutes est présentée par GORDON [Gor95], c'est l'algorithme *fitted-Q*, que nous exposons algorithme 2. Le choix du régresseur de l'étape de la ligne 6 étant laissé libre, *fitted-Q* a donné lieu à de nombreux travaux exploitant cette liberté. Citons par exemple l'utilisation de réseaux de neurones par RIEDMILLER [Rie05] ou d'arbres par ERNST, GEURTS et WEHENKEL [EGW05].

Nous allons nous intéresser à une autre famille, les algorithmes d'itération de la politique [18]. Il s'agit de répéter un cycle consistant à évaluer la fonction de qualité d'une politique puis de remplacer la politique par la politique gloutonne vis-à-vis de la fonction de qualité que l'on vient de calculer.

18. L'idée vient de la PD : [How60]

L'évaluation de la politique peut être effectuée grâce à LSTD de BRADTKE et BARTO [BB96]. LSTD utilise un schéma classique, celui de l'approximation linéaire de la fonction de valeur [19]. On réduit alors le problème du contrôle optimal au choix du vecteur de paramètres ξ donnant la meilleure approximation de la fonction de qualité optimale :

19. On utilise parfois le terme de "fonction de valeur" indistinctement pour la fonction de valeur et la fonction de qualité.

$$\hat{Q}_R^{\pi_R^*}(s,a) = \xi^T \phi_Q(s,a). \tag{2.49}$$

Algorithme 2: Fitted-Q

Entrées : Un ensemble d'entraînement D^R_{sasr};
Sorties : Une politique presque optimale
Initialiser Q arbitrairement ;
pour $t \in [\![1;T]\!]$ **faire**
  Initialiser $D^Q_{saq} \leftarrow \varnothing$;
  pour $(s_i, a_i, s'_i, r_i) \in D^R_{sasr}$ **faire**
    $D^Q_{saq} = D^Q_{saq} \cup \{(s_i, a_i), r_i + \max_{a'} Q(s'_i, a')\}$;
  $Q \leftarrow$ régresseur entraîné sur D^Q_{saq};
retourner $s \rightarrow \arg\max_{a \in \mathcal{A}} Q(s,a)$

Comme pour la classification, le choix des fonctions d'attributs ϕ_Q n'est pas anodin car il va fortement influencer la qualité du contrôle.

Pour trouver les paramètres ξ de la qualité d'une politique π, LSTD minimise (dans le sens des moindres carrés) la distance entre $B^\pi_R \hat{Q}^\pi_R$ et son projeté dans l'espace d'hypothèse induit par ϕ_Q.

L'utilisation de LSTD dans un schéma d'itération de la politique donne l'algorithme d'AR LSPI proposé par Lagoudakis et Parr [LP03] et résumé algorithme 3.

Algorithme 3: LSPI

Entrées : Une base de données D^R_{sasr};
Un critère d'arrêt ϵ;
Données : Un coefficient de régularisation λ
Sorties : Une politique quasi-optimale π;
$\Delta \leftarrow \infty$;
Initialiser ξ arbitrairement;
Initialiser π arbitrairement;
tant que $\Delta > \epsilon$ **faire**
  $\xi' \leftarrow \xi$;
  $\xi \leftarrow (A + \lambda I)^{-1} b$ avec :

$$A = \sum_{(s_i,a_i,s'_i,r_i) \in D^R_{sasr}} \phi_Q(s_i,a_i)\big(\phi_Q(s_i,a_i) - \gamma \phi_Q(s'_i, \pi(s'_i))\big)^T$$

  et

$$b = \sum_{(s_i,a_i,s'_i,r_i) \in D^R_{sasr}} \phi_Q(s_i,a_i) r_i$$

  $\pi \leftarrow g(\pi) : s \rightarrow \arg\max_{a \in \mathcal{A}} \hat{Q}^\pi_R(s,a)$;
  $\Delta \leftarrow \|\xi' - \xi\|_2$;
retourner π

Les algorithmes 2 et 3 n'ont besoin que d'une base de données recueillies une fois pour toutes [20]. Il ne s'agit pas exactement d'une base $D^{\pi^E}_{sa}$ comme la classification en utilise, mais d'une base plus contraignante à réunir, qui contient l'état suivant l'action (afin d'obtenir de l'information sur la dynamique du système) et le signal de récompense (afin d'obtenir de l'information sur la qualité du contrôle) :

$$D^R_{sasr} = \{(s_i, a_i, s'_i, r_i = R(s_i,a_i)) | i \in [\![1;N]\!]\}. \tag{2.50}$$

20. La base D^R_{sasr} devant être représentative de la dynamique du PDM, elle est généralement basée sur une base $D^{\sim}_{sas}$ recueillie par une politique aléatoire à laquelle on ajoute l'information de récompense. Du point de vue des notations, l'exposant n'est donc plus la politique d'échantillonnage comme par exemple dans la base $D^{\pi^E}_{sa} = \{(s_i,a_i) | i \in [\![1;N]\!]\}$ mais la fonction de récompense R telle que $r_i = R(s_i,a_i)$.

Les échantillons de D^R_{sasr} n'ont pas besoin de se suivre, on peut avoir $s_{i+1} \neq s'_i$. Il importe que la base d'échantillons soit représentative de la dynamique du PDM pour chaque action. Une technique pratique et de réunir plusieurs trajectoires d'une politique aléatoire, en la faisant démarrer de différents états.

Les deux algorithmes *fitted-Q* et LSPI sont *batch* et *off-policy*. Cela signifie respectivement qu'ils peuvent utiliser des données échantillonnées une fois pour toutes et n'ont pas besoin de contrôler le système. Ils sont donc plus efficaces en échantillons que les algorithmes en ligne (mais plus coûteux en temps de calcul) et potentiellement plus sûrs aussi, puisque l'on a pas à placer un agent au contrôle du système avant que son entraînement n'ait eu lieu.

2.3 *Définition de l'ARI*

Nous voilà donc armés pour résoudre (de manière approchée) le problème du contrôle optimal, pour peu que l'on dispose d'une description de la tâche à accomplir sous la forme d'une fonction de récompense R et d'échantillons représentatifs du problème.

2.3.1 *Définition du problème*

La question qui se pose maintenant est de savoir dans quelle mesure il est possible d'automatiquement déduire d'une trace du comportement de l'expert une description de son but sous la forme d'une fonction de récompense. Cette méthode d'imitation a été suggérée pour la première fois par RUSSELL [Rus98]. L'état de l'art sera traité au chapitre suivant, nous nous attelons ici à trouver une formulation mathématique du problème et à exposer les notions nécessaires à l'analyse des méthodes existantes et à l'introduction des nouvelles approches que nous proposons.

Outre l'intérêt intrinsèque de la découverte des motivations de l'expert, apprendre une fonction de récompense correspondant au comportement de l'expert permet de soigner les méthodes d'imitation de leur "myopie" (sous-section 2.1.6), en guidant l'agent vers une imitation non pas de la manière dont l'expert accomplit la tâche, comme le fait l'imitation directe de la politique, mais du but de l'expert.

Dans le formalisme des PDM introduit plus tôt, l'expression du problème de l'ARI est de prime abord simple : on suppose que la politique de l'expert π^E est une politique optimale pour une certaine fonction de récompense R^E qu'il s'agit de trouver. Les choses se corsent malheureusement extrêmement rapidement. Tout d'abord, cette formulation n'est pas un problème mathématique bien posé au sens d'HADAMARD [Had02] : il existe en effet de multiples solutions, dont au moins une est dégénérée, la récompense uniformément nulle. Tous les comportements (donc celui de l'expert) ont la même valeur (0) pour cette récompense, donc tous sont optimaux. En inversant, la politique de l'expert est optimale pour la récompense nulle, qui est donc solution du problème.

Plus subtilement, utiliser le formalisme des PDM présuppose que certaines conditions sont réunies, notamment en ce qui concerne les espaces d'état et d'action. Il faut être en mesure de définir un espace d'état markovien, afin qu'un agent puisse prendre sa décision quant à l'action en se basant uniquement sur les informations de l'état. Il faut également être en mesure d'obtenir une trace de l'expert (le problème se pose déjà pour l'imitation directe de la politique). L'espace d'action doit rester discret et de faible cardinal si l'on veut que l'immense majorité des approches d'ARI y restent tractables.

Enfin, il faut que l'expert soit effectivement optimal, l'introduction d'erreurs [21] dans la démonstration pouvant poser problème.

21. Certaines approches d'ARI relâchent la contrainte d'optimalité de l'expert par exemple par l'introduction de variables d'écart.

Ces conditions, qu'il est difficile de satisfaire en totalité, ne vont cependant pas nous préoccuper. Nous nous attelons à trouver de nouvelles solutions au problème formulé par Russell [Rus98]. Pour mesurer notre succès, nous envisageons deux mesures de la qualité de la récompense $\widehat{R}$ trouvée par un algorithme d'ARI. L'une est objective :

$$\mathbb{E}\left[V_{R^E}^{\pi^E}(s) - V_{R^E}^{\pi^*_{\widehat{R}}}(s) \middle| s \sim \rho_E\right], \tag{2.51}$$

l'autre est estimable :

$$\mathbb{E}\left[V_{\widehat{R}}^{\pi^*_{\widehat{R}}} - V_{\widehat{R}}^{\pi^E}(s) \middle| s \sim \rho_E\right]. \tag{2.52}$$

Ces deux critères sont positifs ou au mieux nuls, il s'agit de les minimiser. Les deux termes bleus sont des fonctions de valeur optimales (puisque π^E est optimale pour R^E et que $\pi^*_{\widehat{R}}$ est optimale pour $\widehat{R}$). Ces deux critères sont donc liés à l'optimalité des valeurs marrons et tendront vers 0 quand, pour le premier la politique apprise en optimisant la récompense $\widehat{R}$ est optimale pour la récompense inconnue R^E et quand pour le second la récompense $\widehat{R}$ est telle qu'elle admet la politique expert comme politique optimale.

La distribution en jeu ρ est également importante. La distribution selon laquelle les données de l'expert sont tirées, ρ_E, est celle considérée par défaut, comme c'est le cas pour les méthodes supervisées (voir explication en sous-section 2.1.2). L'ARI présente cependant sur les méthodes supervisées l'avantage de pouvoir bien mieux généraliser aux parties de l'espace d'état sur lesquelles l'expert n'a pas fourni de démonstrations [22], il peut donc être intéressant d'évaluer la qualité du contrôle non pas sur la distribution correspondant aux démonstrations de l'expert, mais justement sur une distribution à support plus large.

22. Cela est principalement dû au fait qu'une fois la récompense trouvée par un algorithme d'ARI, on l'optimise sur tout l'espace d'état grâce à un algorithme d'AR ou de PD.

Le premier critère est celui qui nous intéresse vraiment. Il juge le comportement issu de l'optimisation par un algorithme d'AR ou de PD de la récompense $\widehat{R}$ à l'aune de la description de la tâche confiée à l'expert, à savoir R^E. Cela permet dans les cas où plusieurs bonnes solutions existent de ne pas pénaliser un agent qui aurait fait un choix différent de celui de l'expert si cela importe peu. Il s'agit donc d'une mesure plus fine que l'erreur de classification $\epsilon_C^{\rho_E}$, qui sanctionne toute divergence d'opinion avec l'expert. Lors du test d'algorithmes d'ARI sur des problèmes jouet où la récompense R^E nous est connue, nous étudierons ce critère ou un critère équivalent. Sur de vrais problèmes, en revanche, il est impossible de l'estimer : il faudrait paradoxalement résoudre le problème de l'ARI pour savoir si on a bien résolu le problème de l'ARI.

Le second critère a le défaut d'être optimisé par la récompense nulle. Nous verrons cependant, en analysant les différentes approches d'ARI, les mécanismes mis en place afin d'éviter les solutions dégénérées. Il a l'avantage de pouvoir être estimé puisqu'à l'inverse de R^E, la fonction de récompense $\widehat{R}$ est connue. Un algorithme comme LSTD peut fournir une approximation *off-policy* d'une fonction de valeur, c'est-à-dire évaluer une politique arbitraire, différente de celle qui contrôle le système.

2.3.2 *Attribut moyen*

Le fonction de récompense étant l'objet recherché celle-ci sera, pour ne pas bouder une méthode éprouvée, approximée par un modèle linéaire :

$$R_\theta(s,a) = \theta^T \phi_R(s,a). \tag{2.53}$$

Là encore, le choix des attributs est important. Il faut qu'ils permettent à la fonction de récompense de décrire la tâche effectuée par l'expert.

Dans le contexte de l'approximation linéaire de la fonction de récompense, l'expression de la fonction de valeur d'une politique π fait apparaître un terme que nous appellerons μ^π dont le prochain chapitre révèlera l'importance :

$$V^\pi_{\widehat{R}}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \widehat{R}(s_t, a_t) \middle| s_0 = s, \pi\right] \tag{2.54}$$

$$= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \theta^T \phi_R(s_t, a_t) \middle| s_0 = s, \pi\right] \tag{2.55}$$

$$= \theta^T \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi_R(s_t, a_t) \middle| s_0 = s, \pi\right] \tag{2.56}$$

$$= \theta^T \mu^\pi(s) \tag{2.57}$$

Ce terme, l'attribut moyen d'une politique, est une fonction vectorielle qui porte la structure temporelle du PDM contraint par la politique. C'est, que l'on nous pardonne l'oxymore, une trace du futur qui est liée à la distribution qu'occupera l'agent à partir d'un certain état. La présence du facteur d'amortissement γ dans l'expression empêche l'identification complète à une distribution, ce qui est heureux car sans cela l'expression tendrait vers la distribution stationnaire quel que soit l'état de départ (à la projection dans Φ près), or on souhaite que l'attribut moyen ne soit pas une fonction constante de l'espace d'état-action.

De la même manière que la fonction de qualité donne à la fonction de valeur un degré de liberté supplémentaire, il est possible de donner une action en argument à l'attribut moyen avec le même effet :

$$\mu^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi_R(s_t, a_t) \middle| s_0 = s, \pi\right] \tag{2.58}$$

$$\mu^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi_R(s_t, a_t) \middle| s_0 = s, a_0 = a, \pi\right]. \tag{2.59}$$

L'attribut moyen donne une idée des états que traversera l'agent. Il contient donc des informations sur la dynamique induite par sa politique. Cela va avoir un impact important sur la qualité des diverses approximations linéaires dont nous avons parlé si l'attribut moyen est utilisé en tant qu'attribut tout court, comme il le sera au chapitre 5. La présence de la dynamique du PDM dans l'attribut moyen est illustrée Figure 2.2.

Une propriété immédiate de l'attribut moyen[23] est exploitée par une classe d'algorithmes d'ARI (que nous décrirons plus en détail en section 3.3). Cette propriété découle directement de la définition de l'attribut moyen :

23. Exposée par exemple dans [AN04].

Proposition 1 (Proximité de l'attribut moyen et valeur)**.** *Soit π_1 et π_2 deux politiques sur un même PDM, dont les attributs moyens respectifs μ^{π_1} et μ^{π_2} sont proches (pour la norme $||\cdot||_2$) :*

$$\forall s \in \mathcal{S}, ||\mu^{\pi_1}(s) - \mu^{\pi_2}(s)||_2 \leq \epsilon \in \mathbb{R}. \tag{2.60}$$

Alors quelle que soit la récompense $R_\theta = \theta^T \psi$, les valeurs des deux politiques π_1 et π_2 sont également "proches" :

$$\forall s \in \mathcal{S}, \left|V^{\pi_1}_{R_\theta}(s) - V^{\pi_2}_{R_\theta}(s)\right| \leq ||\theta||_2 \, \epsilon. \tag{2.61}$$

Démonstration. D'après l'Équation 2.57 :

$$\forall s \in \mathcal{S}, \left| V_{R_\theta}^{\pi_1}(s) - V_{R_\theta}^{\pi_2}(s) \right| = \left| \theta^T \mu^{\pi_1}(s) - \theta^T \mu^{\pi_2}(s) \right| \tag{2.62}$$

$$= \left| \theta^T (\mu^{\pi_1}(s) - \mu^{\pi_2}(s)) \right|. \tag{2.63}$$

$$\tag{2.64}$$

En utilisant l'inégalité de Cauchy-Schwartz, on conclut :

$$\forall s \in \mathcal{S}, \left| V_{R_\theta}^{\pi_1}(s) - V_{R_\theta}^{\pi_2}(s) \right| \leq ||\theta||_2 \, \|\mu^{\pi_1}(s) - \mu^{\pi_2}(s)\|_2 \tag{2.65}$$

$$\leq ||\theta||_2 \, \epsilon. \tag{2.66}$$

□

Ce résultat est assez intuitif, si deux politiques induisent les mêmes trajectoires, elles auront même valeur. La manière dont l'apprentissage supervisé résout le problème de l'imitation donne ce type de résultat. Il est intéressant de noter que dans certains cas la réciproque est fausse. Il est donc théoriquement possible de résoudre le problème de l'ARI en s'éloignant des solutions que l'apprentissage supervisé peut proposer. Il est en effet possible d'imaginer qu'un algorithme d'ARI renvoie une récompense qui, une fois optimisée, donnera une politique dont l'attribut moyen est différent de celui de la politique de l'expert, mais de valeur semblable.

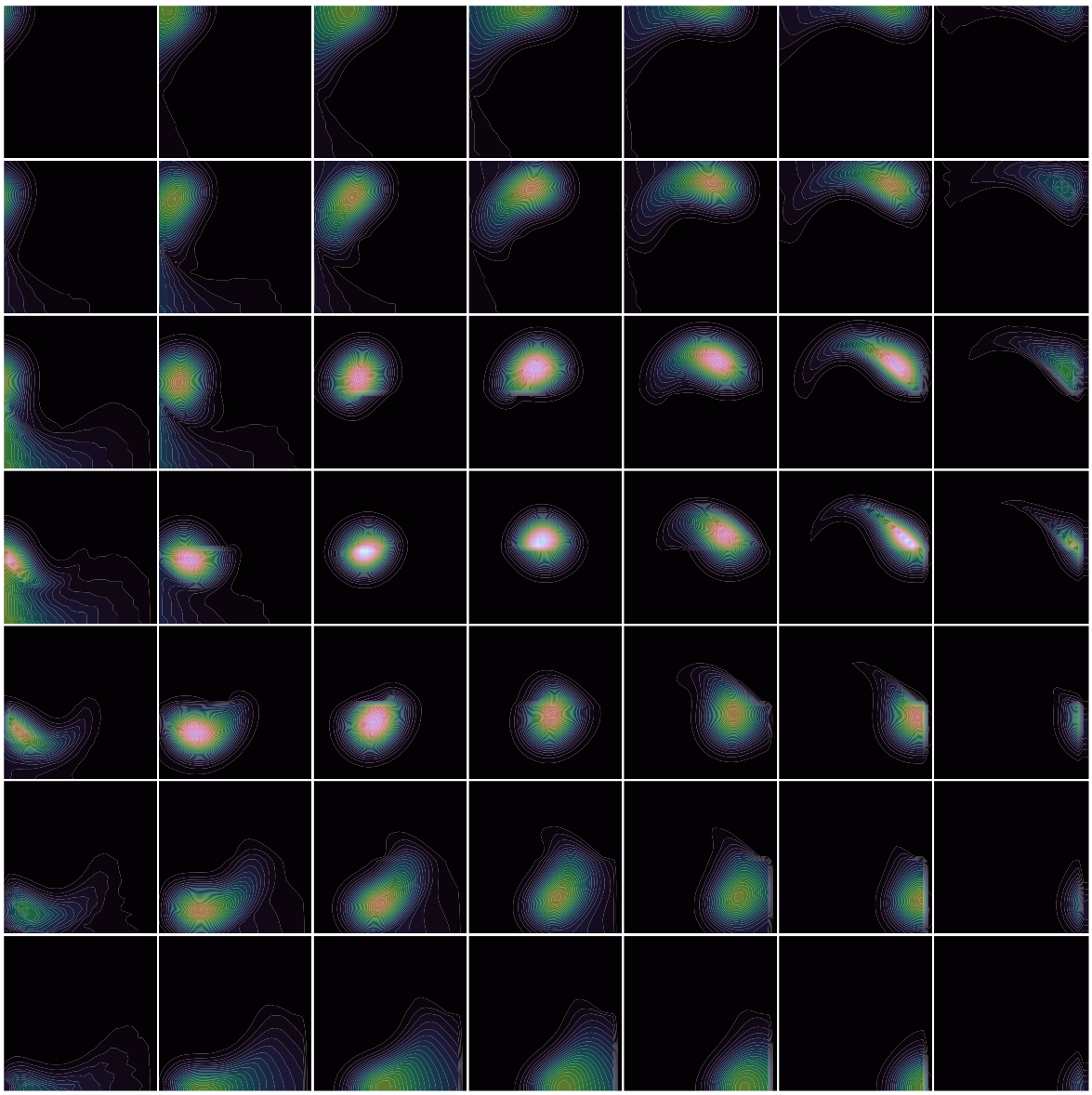


FIGURE 2.2: À comparer à la Figure 2.1 , l'attribut moyen (ici celui de l'expert sur le problème du *mountain-car*) est un reflet de la dynamique du système contraint par la politique. La forme en escargot, caractéristique de trajectoires en va-et-vient à l'apogée de plus en plus haute, a clairement laissé sa trace sur l'attribut moyen.

3
Problématique de l'ARI et état de l'art

Avant d'analyser les approches existantes d'ARI, intéressons nous plus en détail à la définition précise de la fonction de récompense. Nous allons voir quelles transformations il est possible de lui appliquer sans changer les politiques optimales, une technique connue sous le nom de *reward shaping*.

3.1 Fonction de récompense

3.1.1 Espace de départ

En AR, on utilise souvent la transition s, a, s', r comme unité d'information [1]. On y trouve de l'information sur la dynamique du système : l'agent ayant choisi a en s, il est arrivé en s', ainsi que le signal de récompense. Il peut être envisagé d'utiliser une récompense dépendant de la transition complète s, a, s', mais on peut facilement définir une récompense parfaitement équivalente sur $\mathcal{S} \times \mathcal{A}$:

$$R(s,a) = \sum_{s' \in \mathcal{S}} p\left(s'|s,a\right) R'(s,a,s'). \tag{3.1}$$

1. L'algorithme SARSA utilise comme son nom l'indique une séquence augmentée de l'action a' suivant l'état d'arrivée s'.

Une récompense sur les états seuls est envisageable, et permet de transférer une tâche d'un agent à l'autre sans tenir compte de l'espace d'action de chacun des agents, pour peu que tous deux aient le même espace d'état [2]. Cela est cependant problématique car il y a une perte de généralité entre une récompense sur $\mathcal{S}$ et une récompense sur $\mathcal{S} \times \mathcal{A}$ qu'il est impossible de compenser sans changer le PDM [3]. De plus, les récompenses sont généralement définies sur $\mathcal{S} \times \mathcal{A}$ dans la littérature.

Pour toutes ces raisons, nous considèrerons les récompenses comme définies dans $\mathcal{S} \times \mathcal{A}$ par défaut.

2. Par exemple sur un échiquier où la récompense est définie sur l'état (la case où la pièce se trouve) on peut entraîner indistinctement une tour, une reine ou un cavalier à se déplacer à l'endroit voulu bien que ces pièces ne se déplacent pas de la même manière.

3. Par exemple sur un PDM à deux états et deux actions : *rester* et *changer*, il est impossible de définir une récompense sur les états seuls qui admette pour politique optimale celle consistant à toujours choisir l'action *changer*.

3.1.2 Reward shaping

L'étude de la relation entre la condition d'optimalité d'une politique (comme celle de l'expert) et la fonction de récompense associée (que l'on cherche à connaître) nous amène à découvrir d'autres propriétés importantes pour l'analyse du problème de l'ARI. Nous nous intéressons particulièrement aux transformations qu'il est possible d'appliquer à la récompense sans perturber la relation d'ordre existant entre les valeurs des différentes politiques (relation d'ordre sur laquelle est basée la notion d'optimalité Équation 2.35). Nous ne souhaitons pas voir ces transformations en mesure de minimiser la fonction objectif d'un algorithme d'ARI. En effet, cela signifierait que l'on modifie la récompense sans modifier le sens qu'elle porte et donc qu'en réalité on ne résout pas le problème de l'ARI mais une formulation dégénérée de celui-ci.

Un résultat immédiat, qui découle du fait que la condition d'optimalité d'une politique est basée sur une relation d'ordre sur les valeurs, qui sont par définition linéaires vis-à-vis de la récompense, est qu'il est possible d'appliquer n'importe quelle transformation affine de coefficient strictement positif à une récompense sans en changer la signification. Ainsi notamment nous pourrons normaliser une récompense sans perte de généralité.

L'étude des transformations n'ayant pas d'impact sur les politiques optimales a été menée dans le cadre de l'AR. L'objectif est de faciliter la recherche de politiques optimales sans modifier celles-ci. NG, HARADA et RUSSELL [NHR99] trouvent une élégante condition nécessaire et suffisante à la préservation de politiques optimales par changement de récompense : à une récompense R définie sur le triple espace état-action-état $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$, on peut ajouter toute fonction R' définie sous la forme :

$$R'(s, a, s_{+1}) = \gamma R_{\mathcal{S}}(s_{+1}) - R_{\mathcal{S}}(s), \tag{3.2}$$

où $R_{\mathcal{S}}$ est définie sur l'espace d'état. Cette condition est suffisante dans le sens que toute fonction de récompense définie de la sorte peut être ajoutée à une fonction de récompense de $\mathbb{R}^{\mathcal{S}\times\mathcal{A}\times\mathcal{S}}$ sans en changer les politiques optimales. Elle est nécessaire en ceci qu'une fonction de récompense R' pourra être ajoutée à une autre si il existe une fonction $R_{\mathcal{S}}$ qui permette de l'écrire sous la forme donnée en Équation 3.2. Si une telle fonction n'existe pas, il est possible que les probabilités de transition soient telles que la modification de la fonction de récompense entraînera une modification des politiques optimales.

Maintenant que nous disposons d'une meilleure compréhension de l'objet que nous cherchons, voyons comment les approches existantes mènent cette recherche.

3.2 *Premières formulations du problème*

Dans la dernière partie de sa contribution, RUSSELL [Rus98] pose le problème de l'ARI en termes généraux. Il ne propose pas de solution, mais explique l'intérêt du problème et isole des questions-clefs dont certaines restent encore ouvertes aujourd'hui.

C'est dans [NR00] que les premières solutions apparaissent. Afin de trouver la récompense, les auteurs inversent l'équation de Bellman et formulent ainsi une condition nécessaire et suffisante sur la récompense pour qu'elle soit solution du problème de l'ARI. Cette condition n'écarte pas les solutions dégénérées comme la récompense nulle. En conséquence, pour les PDM finis où les probabilités de transition sont connues, les auteurs ajoutent à cette condition nécessaire et suffisante une contrainte de pénalisation du désaccord avec l'expert ainsi qu'une contrainte de régularisation en norme ℓ_1 et obtiennent un programme linéaire. Pour les PDM dont l'espace d'état est grand ou infini, une paramétrisation linéaire de la fonction de récompense est naturellement choisie (rares sont les approches qui choisissent un autre schéma). De fait, et bien qu'il ne porte pas encore ce nom, l'attribut moyen de l'expert apparaît dans le développement. Une version échantillonnée du programme linéaire découlant du schéma d'approximation linéaire de la récompense est proposé. La contrainte d'optimalité repose non plus sur la connaissance des probabilités de transition, mais sur la recherche d'une récompense donnant à la politique de l'expert une plus grande valeur qu'aux politiques optimales trouvées en optimisant la récompense telle que connue pendant les itérations précédentes de l'algorithme (il faut donc résoudre un PDM à chaque itération).

On retrouve dans ce travail plusieurs éléments communs aux approches ultérieures. Il y a le constat primordial que contrairement à ce qui se passe pour l'AR où l'on cherche

le point fixe d'un opérateur contractant (quand on estime une fonction de valeur) et qui donc existe et est unique, on a ici affaire à un problème mal posé, et il convient d'introduire des contraintes supplémentaires afin d'éviter les solutions dégénérées. Le choix et la justification de ces contraintes est un bon moyen de différencier les approches existantes. Il y a le schéma presque ubiquitaire de l'approximation linéaire de la fonction de récompense, qui entraîne l'apparition de l'attribut moyen. Enfin, on retrouve la nécessité pour l'algorithme d'ARI de résoudre le problème de l'AR de manière répétée.

3.3 Méthodes nécessitant la résolution répétée d'un PDM

Cette nécessité est lourde de conséquences quant à l'usage pratique d'un algorithme d'ARI. La résolution du problème de l'AR nécessite en effet soit beaucoup de données pour sa résolution *off-policy*, soit l'accès à un simulateur ou au système réel pour sa résolution *on-policy*.

Abbeel et Ng [AN04] proposent l'algorithme *Projection Inverse Reinforcement Learning* (PIRL) dans un travail séminal dont la structure va être reprise (plus ou moins consciemment) par plusieurs autres approches. La récompense est paramétrée linéairement, de fait la notion d'attribut moyen apparaît assez logiquement (Équation 2.54). Elle obtient même une place centrale dans cette approche itérative de l'ARI. L'idée (résumée algorithme 4) est, à chaque itération, d'apprendre la politique optimale pour la fonction de récompense courante (via un algorithme d'AR). L'attribut moyen de cette politique est alors évalué. La fonction de coût que minimise l'approche joue sur une définition de la proximité d entre l'attribut moyen de cette politique et l'attribut moyen de la politique de l'expert. A chaque itération cette proximité est évaluée, et le vecteur de paramètre θ de la récompense change dans une direction (donnée par une règle de mise à jour u) qui va chercher à réduire cette distance. La justification est que deux politiques d'attributs moyens semblables auront des fonctions de valeur semblables pour toute récompense, y compris donc la récompense inconnue optimisée par l'expert (Proposition 1). Le seul bémol de cette logique est que la réciproque peut être fausse, deux attributs extrêmement différents peuvent mener à la même valeur. Plusieurs autres approches sont basées sur le même schéma itératif. Une revue en est proposée par Neu et Szepesvári [NS09].

PIRL est intrinsèquement un algorithme d'imitation plus que d'ARI, en ce sens que la sortie est une politique stochastique (plus précisément un mélange de politiques déterministes) et non une fonction de récompense. Il est cependant facile de le "convertir" en extrayant la meilleure fonction de récompense, celle liée à la plus petite distance entre attribut moyen de la politique optimale et attribut moyen de l'expert.

La technique proposée dans [NS07] (*Policy Matching* (PM)) repose sur une réduction des désaccords entre la politique de l'expert et la politique à l'itération courante, via une recherche dans l'espace des paramètres θ de la récompense. Le lien unissant récompense et politique, qui est lié à la résolution d'un PDM, est ici caractérisé par des outils d'analyse fonctionnelle afin d'extraire une solution analytique du problème qui soit également calculable. Cette contribution présente une plus grand robustesse aux changements d'échelle des attributs. On a vu en section 3.1 que l'ensemble des politiques optimales est stable par dilatation de la récompense, et donc également par dilatation des attributs. Cette plus grande robustesse est donc signe d'une formulation plus saine de l'ARI. Le bruit dans les attributs est également mieux géré.

Basé sur la théorie des jeux dans un contexte où l'agent définit une politique qui doit être meilleure que celle de l'expert pour une récompense choisie par l'adversaire, l'algo-

Algorithme 4: Structure commune à la plupart des algorithmes d'ARI

Entrées : Une démonstration de la politique experte π^E;
Un critère d'arrêt ϵ;
Données : Une notion de distance d permettant d'évaluer la similarité de deux attributs moyens;
Une règle de mise à jour u permettant de réduire la distance;
Une méthode d'évaluation de l'attribut moyen d'une politique arbitraire;
Une méthode de résolution d'un PDM;
Sorties : Un vecteur de paramètres θ pour la récompense R_θ

```
Calculer l'attribut moyen de l'expert μ^E à partir de la démonstration;
Initialiser θ arbitrairement;
tant que d(μ^E, μ^π̂) > ε faire
    π ← Une politique (quasi) optimale pour la récompense R̂_θ;
    μ^π̂ ← Une approximation de l'attribut moyen de la politique π;
    θ ← u(θ, μ^E, μ^π̂);
retourner θ
```

rithme *Multiplicative Weights for Apprenticeship Learning* (MWAL) de Syed et Schapire [SS08] tombe sur un os[4]. L'inclusion de savoir *a priori* quant à la contribution positive ou néfaste de certains attributs à la réalisation de la tâche donne un algorithme d'ARI dont le résultat peut s'avérer meilleur que l'expert ou, encore plus paradoxal, qui peut fonctionner même en l'absence d'expert (sic). En contrepartie de la nécessité de disposer de ces connaissances *a priori*, il est plus rapide à l'exécution que PIRL et capable de gérer la non optimalité de l'expert. Il exige malheureusement de connaître la dynamique du PDM. La définition de l'ARI comme un problème distinct de l'imitation y gagne en clarté car cette contribution mentionne explicitement les problèmes liés aux raisonnements sur des politiques mixées et non des récompenses, sans pour autant les résoudre. Les auteurs rendent aussi explicites les difficultés soulevées en sous-section 2.3.1 quant à la définition des espaces d'état et d'action, notamment le respect de la propriété de Markov.

4. Os à MWAL.

Ce travail est étendu par Syed, Bowling et Schapire [SBS08] qui, en formulant la résolution du MDP comme un programme linéaire, assainissent la formulation de PIRL et MWAL en leur permettant de raisonner sur une politique stationnaire et sans mélange. En formulant également l'ARI comme un programme linéaire, les auteurs proposent *Linear Programming for Apprenticeship Learning* (LPAL), qui retourne un politique (non une récompense). La formulation de ce programme linéaire met en jeu des grandeurs qui correspondent à l'attribut moyen de politiques optimales pour des récompenses arbitraires, la structure itérative est donc toujours présente.

L'approche de classification à marge structurée décrite en sous-section 2.1.3 et utilisée en [RBS07] pour l'imitation supervisée apparaît également dans [RBZ06]. Plutôt que d'associer une action à un état comme dans l'approche supervisée, l'algorithme *Maximum Margin Planning* (MMP) proposé associe une politique optimale à un PDM. Bien que les auteurs s'en défendent, cette formulation est compatible avec la structure familière des autres algorithmes d'ARI de cette période (algorithme 4), comme le démontrent Neu et Szepesvári [NS09]. La philosophie à la base de l'approche reste cependant différente, et bien que les applications envisagées sont variées et non triviales (voir [RSB09], où elles sont toutes réunies et la méthode de boosting proposée en

[Rat+07] expliquée plus en profondeur), d'autres problèmes apparaissent comme la nécessité de résoudre de multiples PDM de manière tractable, et de formuler le problème du contrôle non pas comme un PDM, mais comme de multiples PDM "compatibles" entre eux [5].

La formulation probabiliste de *Maximum Entropy* (MaxEnt) de Ziebart, Maas, Bagnell et Dey [Zie+08] est intéressante car elle formule un critère (l'entropie) pour choisir entre deux politiques qui jusqu'à présent étaient équivalentes (même valeur ou même attribut moyen). Structurellement, cependant, les différences restent minces avec toujours présentes la résolution répétée d'un MDP et l'estimation de l'attribut moyen de politiques arbitraires.

Curieusement semblables à une approche non officiellement ARI et plus ancienne, [CKO01], les travaux de Ramachandran et Amir [RA07] posent l'ARI comme un problème d'inférence bayésienne où l'on cherche la récompense la plus vraisemblable au vu de la démonstration de l'expert. Il en résulte une formulation plutôt saine de l'ARI permettant par exemple d'incorporer de la connaissance *a priori* sans que cela soit pour autant nécessaire, ou de qualifier le niveau de confiance dans la démonstration de l'expert, qui peut alors être sous-optimal. Des arguments similaires à ceux développés pour MaxEnt permettent de lever les ambiguïtés induites par le fait que l'ARI est un problème mal posé. Ces travaux sont étendus par Lopes, Melo et Montesano [LMM09] qui en extrayant une information d'incertitude proposent un schéma d'échantillonnage interactif selon la même logique, et avec les mêmes avantages que ceux développés par Chernova et Veloso (voir la section 2.1) pour l'approche supervisée (voir sous-section 2.1.6). Bien que cette approche diffère des autres dans l'exposition du raisonnement, elle reste très similaire dans la mise en œuvre, puisque'il faut toujours calculer des politiques optimales et obtenir des échantillons de cette politique (pour calculer des *a posteriori* bayésiens et non plus des attributs moyens).

Signalons finalement les travaux de Jin, Qian et Zhu [JQZ10], qui par l'usage des PG plutôt que d'un schéma linéaire pour l'approximation de la récompense, ainsi que par l'usage d'algorithmes d'AR mettant les PG en jeu [RK04 ; DRP09] permet de ne plus avoir à concevoir des attributs pour appliquer le schéma itératif classique de l'ARI, se basant sur les échantillons comme points supports.

La plupart de ces approches sont résumées dans [NS09]. Ces différentes contributions ont le mérite d'observer le problème sous plusieurs angles, de se placer aux limites du problème (expert non optimal, attributs bruités, etc.) et d'aborder des questions fondamentales (notion de distance entre politiques ou récompenses). Le manque d'harmonisation du domaine de l'ARI (encore jeune) se fait sentir. Chacun redéfinit le problème à sa manière. Toutes ces approches sont malgré cela structurellement très similaires : schéma itératif avec résolution répétée de MDP et approximation de l'attribut moyen de la politique courante puis comparaison à celui de l'expert. Les approches les plus tardives font apparaître les difficultés soulevées par la recherche d'un politique mixée, d'une politique au lieu d'une récompense, de l'absence d'un critère commun, de la résolution répétée du MDP et de l'approximation de l'attribut moyen. Si de bonnes solutions aux deux premiers problèmes sont proposées, les autres soucis ne trouvent en revanche pas encore de réponse.

Outre les applications variées développées au fur et à mesure par Ratliff, Silver et Bagnell [RSB09], il est intéressant de constater l'efficacité des algorithmes de cette veine que Abbeel, Coates et Ng [ACN10] ont démontrée par le contrôle d'un hélicoptère radiocommandé en vol acrobatique. L'ARI ne joue cependant qu'un petit rôle dans cette expérience, où la contribution principale est l'apprentissage du modèle dyna-

5. La résolution se fait avec par exemple A^*, il faut que les actions des différents PDM aient le même type d'effet sur les états.

mique de l'hélicoptère. Ce modèle est ensuite utilisé pour la résolution du problème de l'AR, et donc également de manière répétée lors de la résolution de l'ARI. Développer un simulateur est une tâche lourde.

3.4 Méthodes ne nécessitant pas la résolution répétée d'un PDM

Les problèmes intrinsèquement liés à la structure itérative des premiers algorithmes d'ARI étant identifiés, un regain de créativité a permis l'émergence d'approches hétérogènes ces dernières années.

Un schéma itératif (mais sans lien évident avec le schéma itératif classique de la section précédente) permet à LEVINE, POPOVIC et KOLTUN [LPK10] d'effectuer de la sélection d'attributs tout en apprenant la récompense, à l'aide d'une méthode semblable dans sa motivation au *boosting* opéré dans [Rat+07]. La méthode de LEVINE, POPOVIC et KOLTUN [LPK10] semble cependant plus générique, et l'idée qu'une récompense "simple" est préférable y est enracinée. La tractabilité n'est malheureusement pas rendez-vous.

Présenté comme une approche supervisée de l'imitation, la méthode de MELO et LOPES [ML10] repose sur une notion particulière de distance dans un PDM. Cette distance n'est pas uniquement basée sur la définition de l'espace d'état (comme l'est par exemple la distance euclidienne standard) mais prend en compte la dynamique du PDM afin que deux états semblables pour le problème qui nous préoccupe (permettant d'accéder aux mêmes états en effectuant la même action) aient une distance nulle. Le calcul de cette distance cache la résolution du PDM par des méthodes de PD, on trouve le point fixe d'un opérateur par son application répétée. Bien que la récompense n'apparaisse pas explicitement dans l'exposition de la méthode, le raisonnement qui définit l'échantillonnage interactif et la politique renvoyée par l'algorithme est similaire a ce qui a été proposé par LOPES, MELO et MONTESANO [LMM09].

Raisonnant selon la même logique bayésienne que RAMACHANDRAN et AMIR, mais présentant la particularité de ne pas faire appel à une approximation linéaire de la fonction de récompense, le travail de LEVINE, POPOVIC et KOLTUN [LPK11] utilise des PG dans une approche itérative (appelée *Gaussian Processes Inverse Reinforcement Learning* (GPIRL)) qui apprend en même temps la fonction de récompense et le noyau permettant d'en représenter la structure. Elle souffre d'un problème de tractabilité que les auteurs résolvent en partie en proposant une version échantillonnée sur une partie de l'espace d'état, ce qui induit d'autres problèmes, notamment au niveau de la qualité de la généralisation. Cette approche pourrait facilement être augmentée d'un schéma d'échantillonnage actif en raison de la présence de l'information d'incertitude dans les PG.

Les PG sont aussi présents chez QIAO et BELING [QB11], dans une approche également bayésienne mais beaucoup plus analytique. Il ne s'agit pas simplement de calculer un maximum *a posteriori* sur une classe de fonctions de récompenses : le modèle graphique bayésien est sémantiquement plus riche (et donc malheureusement plus lourd), chaque choix de l'expert rajoute une relation dans un modèle bayésien sur les Q valeurs. Cette approche semble ne pas souffrir des problèmes de généralisation qui pénalisent GPIRL, mais paraît encore moins tractable et est réservée aux PDM discrets.

L'algorithme proposé par DVIJOTHAM et TODOROV [DT10] remplace la résolution du problème de l'AR par l'inversion d'un PDM soluble linéairement. Les résultats obtenus sont aussi bon ou meilleurs que ceux obtenus avec les algorithmes itératifs classiques, et le sont plus rapidement. L'astuce réside dans le fait que pour convertir un PDM

arbitraire en PDM soluble linéairement, il faut connaître les probabilités de transition.

Finalement, RelEnt de BOULARIAS, KOBER et PETERS [BKP11] reste dans la lignée des algorithmes itératifs classiques en ceci que son argumentaire ressemble à celui qui amène MaxEnt et qu'il tient compte des progrès fait dans la définition du problème de l'ARI : il s'agit de trouver une récompense (non une politique), sans accès complet à la dynamique du système (mais en ayant accès à des trajectoires) et en observant la politique de l'expert (sans la connaître totalement). La méthode cependant est plus efficace, l'échantillonnage préférentiel permet à l'algorithme de fonctionner en utilisant uniquement des échantillons de l'expert et des échantillons obtenus grâce à une politique aléatoire. La résolution répétée du problème de l'AR n'est plus nécessaire. L'approche échantillonnée est décrite algorithme 5. C'est cette approche, la plus efficace de l'état de l'art en terme d'information nécessaire à son fonctionnement, qui nous servira de point de comparaison lorsque nous testerons empiriquement nos approches des chapitres 5 et 6.

Algorithme 5: RelEnt en version échantillonnée

Entrées : Une base de données $D_{sas}^{\pi^E}$ organisée en M trajectoires de longueurs $L_m, 1 \leq m \leq M$;
Une base de données échantillonnées aléatoirement $D_{sas}^{\sim}$ organisée en M' trajectoires de longueurs L'_m;
Sorties : Un vecteur de paramètres θ pour la récompense $R_\theta = \theta^T \phi$

1 Initialiser θ arbitrairement;

2 Calculer

$$\bar{\mu^E} = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{L_m - 1} \gamma^t \phi(s_{t,m}, a_{t,m});$$

Calculer

$${}^m\mu = \sum_{t=0}^{L'_m - 1} \gamma^t \phi(s_{t,m}, a_{t,m})$$

pour chaque $m \in [\![1; M']\!]$;

3 Procéder à la descente de sous-gradient dont le sous-gradient est :

$$\nabla_\theta = \bar{\mu^E} - \frac{\sum_{m=1}^{M'} \exp(\theta^T({}^m\mu))({}^m\mu)}{\sum_{m=1}^{M'} \exp(\theta^T({}^m\mu))} - E$$

avec E le vecteur de terme général :

$$E_j = \text{sign}(\theta_j)\epsilon$$

où ϵ est un méta-paramètre à régler;

4 **retourner** θ

Ces nouvelles approches abandonnent la structure itérative classique, évitant ainsi les problèmes qu'elle implique, à savoir rendre l'algorithme lent et gourmand en données, et d'un point de vue théorique faire apparaître des termes d'erreur mal maîtrisés.

4
Calcul de l'attribut moyen : LSTD-μ

De par la sémantique dont il est porteur, illustrée Figure 2.2, et de par le rôle qu'il tient dans les algorithmes itératifs énumérés en section 3.3, l'attribut moyen est une notion centrale en ARI.

Nous cherchons à résoudre le problème de l'ARI dans un cadre où la collecte de données est difficile. Cela signifie qu'un simulateur n'est pas disponible. Or un simulateur est nécessaire à l'application de la méthode de Monte-Carlo (algorithme 6) que les méthodes actuelles utilisent. Il nous faut donc trouver une nouvelle méthode d'approximation de l'attribut moyen, c'est la première (chronologiquement) contribution de cette thèse, parue dans [KGP11c ; KGP11a ; KGP11b]. Cette méthode est exposée (ainsi que deux alternatives) section 4.1. La pertinence de cette méthode dans le cadre ouvert par les approches récentes de l'ARI est finalement étudiée section 4.3, qui présage des défis relevés par les contributions des chapitres 5 et 6.

4.1 Principe

Les approches discutées en section 3.3 font toutes appel, du fait de leur structure commune (algorithme 4), à une procédure d'estimation de l'attribut moyen de l'expert ainsi que, de manière répétée, à une procédure d'estimation de l'attribut moyen [1] d'une politique arbitraire π :

$$\mu^\pi \in \left(\mathbb{R}^{d_{\psi_R}}\right)^{\mathcal{S}\times\mathcal{A}} \tag{4.1}$$

$$\mu^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\phi_R(s_t,a_t)\,\middle|\, s_0 = s, a_0 = a, \pi\right] \tag{4.2}$$

1. Souvent, c'est un terme plus simple à estimer, $\mathbb{E}\left[\mu^\pi(s)|\, s\sim\rho_0\right]$ où ρ_0 est par exemple une distribution des états de départ, qu'il est nécessaire d'approximer.

La méthode préconisée pour le calcul de l'attribut moyen est l'utilisation d'un simulateur pour obtenir les échantillons nécessaires à une estimation de Monte-Carlo comme cela est formalisé dans l'algorithme 6. Cet estimateur non biaisé possède de surcroît l'avantage d'être simple à mettre en œuvre… pour peu que l'on dispose d'un simulateur.

L'horizon L après lequel on tronque la trajectoire peut être choisi en fonction du facteur d'amortissement γ : au bout d'un moment le terme γ^t en facteur de l'attribut devient négligeable, poursuivre le calcul de la somme n'a donc plus d'intérêt [2]. Le nombre de trajectoires M est le fruit d'un compromis entre variance et temps de calcul.

2. A noter que la somme $\sum_{t=0}^{\infty}\gamma^t$ converge vers un réel (en l'occurrence $\frac{1}{1-\gamma}$).

Il est à noter que la méthode est sans biais (au fait de tronquer les trajectoires près), pour peu que le simulateur soit fidèle à la réalité. Dans le cas contraire, on obtient un estimateur non biaisé d'une grandeur ne correspondant pas à ce que l'on cherche, c'est le biais de modélisation. Il reste toujours la possibilité d'utiliser des données en

provenance d'un système réel, mais cela implique de pouvoir placer le système dans un état arbitraire et de laisser une politique potentiellement très mauvaise aux manettes sans provoquer de casse, le tout pour un coût en temps ou ressources raisonnable.

Algorithme 6: Estimation de μ^π par une méthode de Monte-Carlo

Entrées : Le couple (s, a) en lequel on souhaite évaluer μ^π;
Données : La politique π et un simulateur du système à contrôler;
Résultat : $\mu^\pi(s,a)$;

$D^\pi_{sa} \leftarrow \varnothing$;
répéter M **fois**
  Simuler une trajectoire D de longueur L, commençant par s et a, puis suivant π par la suite;
  $D^\pi_{sa} \leftarrow D^\pi_{sa} \cup D$;
retourner

$$\frac{1}{M}\sum_{i=1}^{M}\sum_{t=0}^{L-1}\gamma^t \phi_R(s_{t,i}, a_{t,i})$$

Si l'on compare la définition de l'attribut moyen (Équation 2.59) que nous venons de rappeler à la définition de la fonction de valeur (Équation 2.32), on constate que les deux expressions sont quasiment identiques. La fonction de valeur est définie par l'espérance d'un cumul γ-pondéré de récompenses, tandis que l'attribut moyen est l'espérance d'un cumul γ-pondéré d'attributs. La sémantique des ces deux objets n'est pas la même puisque l'attribut ne porte généralement pas d'information immédiatement exploitable quant à la qualité du contrôle mais, mathématiquement, les différences sont minimes. La seule nuance entre l'attribut moyen et la fonction de valeur est la dimension. La fonction de valeur travaille sur une récompense réelle tandis ce qui est sommé dans l'attribut moyen est une grandeur vectorielle (de dimension d_{ϕ_R}).

Si l'on considère l'attribut moyen composante par composante, alors cette dissimilitude s'estompe. La composante j de l'attribut moyen est exactement le même objet que la fonction de valeur pour la récompense donnée par la $j^{\text{ième}}$ composante de l'attribut ϕ_R :

$$\mu^\pi{}_j = V^\pi_{\phi_{R_j}}. \tag{4.3}$$

Sachant cela, il paraît naturel, pour calculer l'attribut moyen d'une politique, d'adapter les techniques existantes permettant de calculer la valeur de cette politique. Nous n'aurons, comme nous allons le voir, même pas à utiliser la décomposition en composante comme nous venons de le faire, le fait que l'expression soit vectorielle plutôt que scalaire ne posant pas de problèmes insurmontables.

Dans le cas idyllique où l'espace d'état est discret de faible cardinal et la matrice des probabilités de transition P^π induites par la politique π est connue, un algorithme d'évaluation de la politique tiré de la PD convient tout à fait à l'estimation de l'attribut moyen, comme l'explicite l'algorithme 7.

Maintenant que le principe d'estimer l'attribut moyen par une méthode de calcul de la fonction de valeur est établi, voyons comment il est possible d'adapter non plus un algorithme de PD mais un algorithme d'AR. La PD suppose connues les probabilités de transition. Cela est en général difficile à réaliser sur les systèmes réels, notamment les systèmes robotiques. Le passage à un algorithme d'AR permettrait d'apprendre l'attribut moyen au travers d'interactions avec le système réel, en minimisant le travail

Algorithme 7: Estimation de μ^π par une méthode de PD

Entrées : La matrice P^π des probabilités de transition $p(s'|s, \pi(s))$;
La fonction d'attributs ϕ_R;
Résultat : La valeur exacte de μ^π
1 Pour chaque composante j de μ^π :

$$\mu^\pi{}_j \leftarrow (I - \gamma P^\pi)^{-1}(\phi_R)_j$$

retourner μ^π

d'ingénierie.

Un algorithme d'estimation de la fonction de valeur particulièrement approprié est LSTD de Bradtke et Barto [BB96]. Dans sa variante proposée par Lagoudakis et Parr [LP03] (voir algorithme 3), LSTD-Q, il permet d'estimer la fonction de qualité d'une politique de manière *batch* et *off-policy*. Cela signifie que le fonctionnement de l'algorithme est complètement découplé de la commande du système. LSTD-Q peut en effet fonctionner à partir de données recueillies une fois pour toutes (c'est l'aspect *batch*, ou *par lots*) par une politique qui n'est pas celle que l'on évalue (c'est l'aspect *off-policy*).

Cet algorithme, comme beaucoup d'autres en AR repose sur une approximation linéaire de la fonction de qualité (Équation 2.49). Il va donc nous falloir utiliser un schéma d'approximation linéaire pour estimer l'attribut moyen qui lui même est issu de la définition d'un schéma d'approximation linéaire pour la récompense[3] (Équation 2.54). On suppose ici que la récompense est approximée selon le schéma de l'Équation 2.53 :

3. Yo dawg, I heard you like linear approximation schemes...

$$R_\theta(s, a) = \theta^T \phi_R(s, a). \tag{4.4}$$

L'attribut moyen est donc l'espérance du cumul γ-pondéré de l'attribut ϕ_R. Nous allons l'approximer dans un espace d'hypothèse engendré par un autre attribut : ϕ_μ :

$$\hat{\mu}^\pi_\zeta(s, a) = \zeta^T \phi_\mu(s, a). \tag{4.5}$$

Ceci posé, l'adaptation de l'algorithme est relativement simple, le calcul de la matrice de paramètres ζ est décrit algorithme 8. La principale différence avec LSTD-Q réside dans la dimension du vecteur de paramètres. Pour LSTD-Q, il s'agit d'un vecteur de dimension d_{ϕ_V}, tandis que pour LSTD-μ, il s'agit d'une matrice de taille $d_{\phi_\mu} \times d_{\phi_R}$. Pour l'évaluation de $\hat{\mu}^\pi_\zeta$ en un couple (s, a), il suffit d'utiliser l'Équation 4.5. L'inversion de matrice au cœur de l'algorithme est la même pour toutes les composantes du vecteur d'attribut moyen, il n'y a donc à la faire qu'une seule fois. La complexité algorithmique de LSTD-μ est donc (asymptotiquement) la même que celle de LSTD-Q.

4.2 *Erreurs d'approximation*

Lazaric, Ghavamzadeh et Munos [LGM10] proposent une borne sur la performance de LSTD lorsque celui-ci est employé pour l'estimation d'un fonction de valeur à l'aide d'une base de N échantillons. Avec probabilité $1 - \delta$, on a :

$$\|\hat{V}^\pi_R - V^\pi_R\|_N \leq \epsilon_{\phi_V} + O\left(\sqrt{\frac{d_{\phi_V} \ln(\frac{d_{\phi_V}}{\delta})}{N}}\right) \tag{4.6}$$

$$\text{avec :} \|f\|_N = \frac{1}{N}\sum_{t=1}^{N} f(s_t)^2 \tag{4.7}$$

Algorithme 8: Estimation de μ^π par LSTD-μ

Entrées : Un base de données $D^{\phi_R}_{sasr}$;
Une fonction d'attribut ϕ_μ;
La politique π;
Un paramètre de régularisation λ;
Résultat : La matrice ζ de paramètres pour l'approximation linéaire de l'attribut moyen;

$A \leftarrow 0$; `/* Matrice de taille` $d_{\phi_\mu} \times d_{\phi_\mu}$ `*/`
$b \leftarrow 0$; `/* Matrice de taille` $d_{\phi_\mu} \times d_{\phi_R}$ `*/`
pour chaque $s, a, s', \phi_R(s,a) \in D^{\phi_R}_{sasr}$ **faire**
  $A \leftarrow A + \phi_\mu(s,a)\big(\phi_\mu(s,a) - \gamma\phi_\mu(s', \pi(s'))\big)^T$;
  $b \leftarrow b + \phi_\mu(s,a)\phi_R(s)^T$;
$\zeta \leftarrow (A + \lambda I)^{-1} b$;
retourner ζ

Chaque composante de l'attribut moyen étant exactement une fonction de valeur (Équation 4.3) cette borne s'applique donc directement. La différence entre la valeur recherchée $V^\pi_{\phi_{R_j}}$ et la valeur trouvée par LSTD-μ $(\hat{\mu}^\pi_\zeta)_j$ est donc bornée par le terme d'erreur relatif au biais inductif ϵ_{ϕ_μ} qui dépend de la différence entre la projection empirique de $V^\pi_{\phi_{R_j}}$ dans l'espace d'hypothèse induit par ϕ_μ et $V^\pi_{\phi_{R_j}}$ elle-même. À cela s'ajoute un terme dépendant de la taille N de la base d'entraînement.

$$\|V^\pi_{\phi_{R_j}} - (\hat{\mu}^\pi_\zeta)_j\|_N = \|\mu^\pi_j - (\hat{\mu}^\pi_\zeta)_j\|_N \leq \epsilon^j_{\phi_\mu} + O\left(\sqrt{\frac{d_{\phi_\mu} \ln(\frac{d_{\phi_\mu}}{\delta})}{N}}\right) \tag{4.8}$$

En appliquant ce résultat à chacune des d_{ϕ_R} composantes de la fonction vectorielle μ^π avec probabilité $1 - \delta'$, nous pourrions borner l'écart maximum avec un probabilité $1 - d_{\phi_R}\delta'$ (application d'une borne d'union). En posant $\delta' = \frac{\delta}{d_{\phi_R}}$, on obtient donc avec probabilité $1 - \delta$:

$$\|\hat{\mu}^\pi_\zeta - \mu^\pi\| \leq \max_{j \in [\![1, d_{\phi_R}]\!]} \left[\epsilon^j_{\phi_\mu}\right] + O\left(\sqrt{\frac{d_{\phi_\mu} \ln(\frac{d_{\phi_\mu} d_{\phi_R}}{\delta})}{N}}\right), \tag{4.9}$$

$$\text{avec : } \|\mu\| = \max_{j \in [\![1, d_{\phi_R}]\!]} \|\mu_j\|_N. \tag{4.10}$$

Lors des évaluations empiriques des algorithmes que nous proposons, nous prendrons :

$$\phi_\mu = \phi_R. \tag{4.11}$$

Afin de ne pas fournir d'information *ad-hoc* aux problèmes sur lesquels nous testons nos approches, nous utiliserons les attributs standards définis équations 2.9 et 2.12.

Pour minimiser les termes d'erreurs $\epsilon^j_{\phi_\mu}$ il est cependant possible de réfléchir à un choix d'attributs plus performants. Il est même possible d'estimer chaque composante de μ^π séparément, et donc de choisir des fonctions d'attribut différentes pour chaque composante. Les autres termes d'erreur diminuent lorsqu'on augmente la taille N de la base d'entraînement.

Cette borne portant sur les échantillons disponibles pour calculer l'approximation, elle est particulièrement adaptée à l'usage que nous ferons de LSTD-μ au chapitre 5,

lorsqu'il sera utilisé pour fournir à SCIRL une estimation de l'attribut moyen de l'expert sur les données que celui-ci a échantillonné.

Une borne en généralisation a également été fournie par LAZARIC, GHAVAMZADEH et MUNOS. Elle s'applique lorsque la chaine de Markov issue du PDM contraint par la politique que l'on souhaite évaluer admet une distribution stationnaire ρ et que les échantillons sont acquis après que le régime stationnaire ait été atteint.

$$\|\hat{V}_R^\pi - V_R^\pi\|_\rho \leq \epsilon_{\phi_V}^\rho + O\left(\sqrt{\frac{d_{\phi_V} \ln(\frac{N d_{\phi_V}}{\delta})}{N}}\right) \tag{4.12}$$

Là encore, il est possible d'étendre cette borne à l'attribut moyen :

$$\|\hat{\mu}_\zeta^\pi - \mu^\pi\| \leq \max_{j \in [\![1, d_{\phi_R}]\!]} \left[\epsilon_{\phi_\mu}^{\rho,j}\right] + O\left(\sqrt{\frac{d_{\phi_\mu} \ln(\frac{N d_{\phi_\mu} d_{\phi_R}}{\delta})}{N}}\right), \tag{4.13}$$

$$\text{avec : } \|\mu\| = \max_{j \in [\![1, d_{\phi_R}]\!]} \|\mu_j\|_\rho. \tag{4.14}$$

Les différences principales avec la borne de l'Équation 4.9 sont l'utilisation de la norme sur la distribution ρ et la présence du terme ϵ_ϕ^ρ qui dépend non plus de la projection empirique de l'objet recherché sur l'espace d'hypothèses, mais simplement de sa projection.

4.3 *Avantages dans le cadre des approches existantes*

Dans le schéma itératif standard des premiers algorithmes d'ARI (algorithme 4), les problèmes techniques se cachent dans les étapes des lignes 4 et 5, à savoir la résolution du PDM et l'estimation de l'attribut moyen. Au delà de l'aspect computationnel, qui pose de moins en moins de problèmes à mesure que le temps passe et que croît la puissance des machines, le goulot d'étranglement réside dans la collecte des données nécessaires à la résolution de ces deux étapes.

Les auteurs des approches mentionnées en section 3.3 présupposent la disponibilité d'un simulateur. Dans bien des cas, la conception de ce simulateur représente un défi. Par exemple dans les travaux déjà mentionnés de ABBEEL, COATES et NG [ACN10], l'apprentissage du modèle est si peu évident qu'il est l'objet principal de la contribution.

Dans les cas où il est impossible ou incommode de construire un tel simulateur, recueillir des données reste cependant souvent faisable. Nous nous plaçons dans le cas où l'expert peut contrôler le système, amassant ainsi une démonstration experte $D_{sas}^{\pi^E}$ tout en pouvant aller explorer d'autres zones de l'espace d'état que celles dans lesquelles il souhaite se trouver, soit que l'on perturbe sa démonstration de manière aléatoire, soit qu'il agisse volontairement de manière sous optimale dans un but d'exploration. Ces données non expertes sont recueillies dans une trace $D_{sas}^{\sim}$. Il s'agit là d'une structure extrêmement contrainte, mais qui correspond par exemple aux problèmes de locomotion robotique ou d'Interaction Homme-Machine où les modèles sont imparfaits et la collecte de données coûteuse.

Dans le cadre de l'AR, l'approche de LAGOUDAKIS et PARR [LP03] peut fonctionner sous de telles contraintes. Elle découple, comme nous l'avons dit sections 2.2.3 et 4.1, la collecte de données et la résolution approchée du PDM. L'aspect de traitement par lot, couplé à l'aspect *off-policy*, permet de réutiliser tous les échantillons pour évaluer une

politique différente de celle qui était au commandes du système lors de la collecte. L'algorithme LSPI, qui utilise *LSTD-Q* dans un schéma d'itération de la politique permet donc de trouver la solution approchée d'un PDM à partir d'une trace D^R_{sasr} [4]. Comme, pour créer LSTD-μ, nous nous sommes inspirés de LSTD-Q, les mêmes avantages sont transférés à l'estimation de l'attribut moyen. Nous pouvons donc facilement imaginer porter le schéma itératif classique de l'ARI décrit dans l'algorithme 4 à un cadre où la collecte de données est coûteuse. C'est ce que présente l'algorithme 9.

4. On rappelle que $D^R{}_{sasr} = \{(s_i, a_i, s'_i, r_i = R(s_i, a_i))|i \in [\![1; N]\!]\}$

Algorithme 9: Instanciation du schéma itératif de l'ARI avec des méthodes LSTD (en bleu les changements par rapport à l'algorithme 4)

Entrées : Une démonstration $D^{\pi^E}_{sas}$ de la politique expert π^E;
Un critère d'arrêt ϵ;
Données : Une notion de distance d permettant d'évaluer la similarité de deux attributs moyens;
Une règle de mise à jour u permettant de réduire cette distance;
Des données non-expertes $D^{\sim}_{sas}$;
Sorties : Un vecteur de paramètres θ pour la récompense $\widehat{R}_\theta$

1 Calculer l'attribut moyen de l'expert μ^E à partir de la démonstration;
2 Initialiser θ arbitrairement;
3 **tant que** $d(\mu^E, \hat{\mu^\pi}) > \epsilon$ **faire**
4 Construire la trace $D^{\hat{R}_\theta}_{sasr}$ à partir de $D^{\sim}_{sas}$ et $\hat{R}_\theta$;
5 Construire la trace $D^{\phi_R}_{sasr}$ à partir de $D^{\sim}_{sas}$ et ϕ_R;
6 $\pi \leftarrow LSPI(D^{\hat{R}_\theta}_{sasr})$;
7 $\hat{\mu^\pi} \leftarrow LSTD\mu(D^{\phi_R}_{sasr})$;
8 $\theta \leftarrow u(\theta, \mu^E, \hat{\mu^\pi})$;
9 **retourner** θ

L'estimation de l'attribut moyen par LSTD-μ est imparfaite, comme nous l'avons étudié section 4.2. Une erreur trop grande empêcherait l'algorithme d'ARI de converger. Dans le schéma de l'algorithme 9, les sources d'erreurs principales sont l'estimation de l'attribut moyen par LSTD-μ et la résolution du PDM par LSPI. Les erreurs introduites par ces deux routines dépendent largement de la qualité et de la quantité des données contenues dans la trace $D^{\sim}_{sas}$ de données non expertes. Elles dépendent aussi bien sûr du choix des fonctions d'attribut ϕ_R et ϕ_μ.

Intuitivement, on peut argumenter que le schéma d'itération de la politique qu'est LSPI, qui implique donc l'évaluation répétée de politiques arbitraires, sera plus problématique qu'une simple évaluation de la politique (même sur plusieurs composantes) comme LSTD-μ. Une expérience permet de confirmer empiriquement cette intuition.

4.4 *Validation empirique*

Cette expérience fut pour la première fois présentée dans notre article [KGP11c]. Le schéma en jeu est celui proposé dans l'algorithme 9 avec la notion de distance d et la règle de mise à jour u choisis conformément aux travaux de Abbeel et Ng [AN04]. L'idée est de comparer ce schéma à un autre qui, toutes choses étant égales par ailleurs, fait appel à un oracle pour l'estimation de $\hat{\mu^\pi}$. La comparaison des deux approches va porter sur leur comportement en fonction de la taille de la trace non experte $D^{\sim}_{sas}$. Lorsque cette trace ne contient pas assez de données, LSPI ne peut résoudre le PDM

et LSTD-μ ne peut estimer correctement l'attribut moyen. Au fur et à mesure que la taille de $D_{sas}^{\sim}$ augmente, LSPI et LSTD-μ seront de mieux en mieux à même de remplir leur mission. Si le besoin en données est le même pour LSTD-μ et LSPI, la version de l'algorithme d'ARI qui profite d'un oracle pour l'attribut moyen devrait voir ses performances augmenter plus vite en fonction de la taille de $D_{sas}^{\sim}$ que la version utilisant LSTD-μ, celui-ci étant censé introduire des erreurs significatives. Si, à l'inverse, les performances des deux versions s'avèrent similaires, cela signifie que les besoins en données de LSPI sont supérieurs à ceux de LSTD-μ. Quand LSPI parvient à résoudre le PDM, LSTD-μ a déjà suffisamment d'information à sa disposition pour fournir une estimation $\hat{\mu}_{\zeta}^{\pi}$ de l'attribut moyen aussi bonne (pour l'usage que l'on veut en faire) que celle de l'oracle.

Le problème jouet (le pendule inversé, section B.1) sur lequel ces deux approches sont testées permettant de définir un simulateur, l'oracle est un estimateur de Monte-Carlo tel que décrit par l'algorithme 6 dont les paramètres L et M sont réglés de manière à obtenir une variance négligeable.

L'expert est la politique apprise par LSPI lorsque celui-ci est entraîné sur une récompense nulle partout, mais négative lorsque le pendule tombe. La paramétrisation ϕ_Q de LSPI est celle suggérée par Lagoudakis et Parr [LP03][5], à savoir un réseau de 3×3 gaussiennes. Afin de minimiser le travail d'ingénierie et de ne pas introduire de savoir *a priori* en utilisant des attributs porteurs de sens, on utilise les mêmes attributs pour la paramétrisation de la récompense et de l'attribut moyen :

5. Le pendule inversé est en effet l'un des bancs de test dans la contribution qui introduit LSPI.

$$\phi_Q = \phi_R = \phi_\mu. \tag{4.15}$$

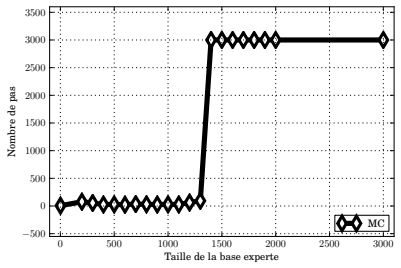
La trace non experte $D_{sas}^{\sim}$ provient d'une politique aléatoire mise au contrôle du pendule alors qu'il se trouve dans une position proche de l'équilibre, jusqu'à ce qu'il tombe. On commence alors un nouvel épisode, jusqu'à ce que la quantité d'échantillons voulue soit atteinte. L'expert fournit 100 échantillons (soit 10 secondes de démonstration) dans la base $D_{sas}^{\pi^E}$. Cela est suffisant pour obtenir une bonne approximation de μ^E par Monte-Carlo. L'expérience est répétée 100 fois afin d'obtenir des informations de variance.

Le résultat de l'expérience est donné Figure 4.1 . On constate que la taille de $D_{sas}^{\sim}$ [6]est bien le facteur déterminant de la qualité de l'imitation. On constate surtout que les deux approches manifestent le même comportement. Cela pourrait signifier que quelle qu'ait pu être l'erreur introduite par l'utilisation de LSTD-μ, elle est gommée par les besoins de LSPI en données. Quand ce dernier dispose de suffisamment de données pour résoudre le PDM suffisamment bien pour que l'algorithme d'ARI converge vers une bonne solution, LSTD-μ possède assez d'information pour fournir une estimation de l'attribut moyen qui, pour l'usage qui en est fait, est aussi bonne que celle de l'oracle.

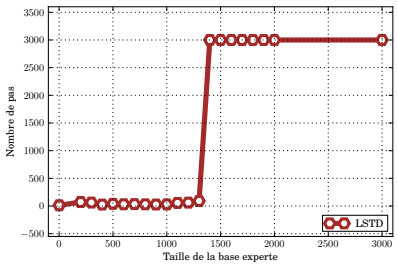
6. La bonne répartition des données dans l'espace d'état-action est également importante. Ici, grâce à l'usage d'une politique aléatoire, $D_{sas}^{\sim}$ est bien représentative de la dynamique.

Le goulot d'étranglement des approches itératives est donc bien la résolution répétée du PDM, il importe de s'affranchir de cette contrainte si l'on souhaite pouvoir utiliser l'ARI sur des problèmes complexes sans avoir à déployer des trésors d'ingénierie.

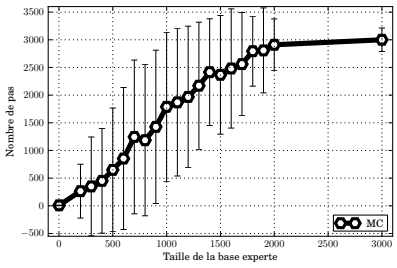
Un algorithme de ce type est l'approche de Boularias, Kober et Peters [BKP11], RelEnt, qui peut fonctionner dans le même cadre contraint que celui que nous venons de décrire, à savoir dans le cadre où les seules données dont on dispose sont une trace de l'expert $D_{sas}^{\pi^E}$ et une trace non experte $D_{sas}^{\sim}$. Il serait cependant souhaitable d'aller encore plus loin et de pouvoir résoudre le problème de l'ARI armé uniquement de données expertes. Ce sont en effet les données les plus naturelles à rassembler.



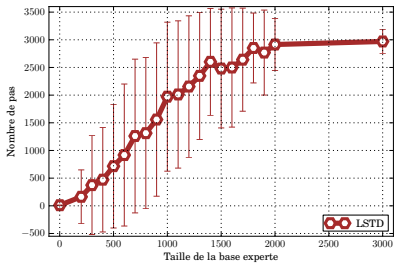
(a) Oracle : une seule expérience.



(b) LSTD-μ : une seule expérience.



(c) Oracle : moyenne et écart-type sur 100 expériences.



(d) LSTD-μ : moyenne et écart-type sur 100 expériences.

FIGURE 4.1: Comparaison de deux approches itératives de l'ARI sur le problème du pendule inversé. L'ordonnée est le nombre de pas de temps durant lesquels la politique évaluée est en mesure de maintenir le pendule en équilibre, avec un maximum à 3000. L'une est décrite algorithme 9 (LSTD sur les courbes), l'autre est identique sauf en ce qui concerne l'estimation de l'attribut moyen, qui est faite par un oracle (MC sur les courbes). On constate que les deux approches ont le même comportement. Dès lors que la quantité de données obtenues aléatoirement est suffisante pour que LSPI résolve le PDM correctement, l'algorithme d'ARI converge vers une bonne solution. Cela démontre empiriquement que les erreurs introduites par la résolution répétée du PDM sont plus importantes que celles liées à l'estimation de l'attribut moyen par notre algorithme LSTD-μ.

4.5 *Conclusion*

L'algorithme LSTD-μ que nous proposons nécessite pour fonctionner moins de données que l'algorithme d'AR LSPI. Les garanties théoriques applicables à LSTD se transfèrent à LSTD-μ. De manière générale, on peut imaginer adapter d'autres résultats de LSTD à LSTD-μ, comme par exemple l'utilisation de traces d'éligibilité [SG11], l'utilisation de projections aléatoires [Gha+10] ou l'introduction de méthodes parcimonieuses [KN09 ; Gei+12], entre autres.

5
Classification structurée pour l'apprentissage par renforcement inverse

Comme nous venons de le voir sur l'exemple précédent, c'est bien la résolution répétée du PDM qui rend les algorithmes d'ARI itératifs (algorithme 4) si gourmands en information. Il faut donc trouver un moyen de résoudre le problème de l'ARI sans avoir à résoudre de PDM. Parmi les algorithmes les plus récents (section 3.4), la plupart ont besoin de données qui permettraient de résoudre le PDM, et ne présentent donc pas, du point de vue de la facilité d'utilisation dans un cadre où la collecte de données est difficile, un avantage décisif sur les approches itératives.

Un de ces algorithmes, cependant, sort du lot puisqu'il ne nécessite pas la résolution du PDM et peut fonctionner à l'aide de simplement deux bases d'entraînement $D_{sas}^{\pi^E}$ et $D_{sas}^{\sim}$ comme notre algorithme 9. Il s'agit de RelEnt. Comme il ne nécessite ni de connaître les probabilités de transition, ni de résoudre un PDM de manière répétée, il sera celui auquel nous comparerons les approches définies dans ce chapitre et le suivant, approches qui constituent la contribution principale de cette thèse et qui sont (à notre connaissance) les seules capables de résoudre le problème de l'ARI uniquement grâce à des données expertes [1].

1. Nonobstant l'utilisation d'heuristiques.

5.1 Liens entre classification et ARI

Les approches supervisées d'imitation (sous-section 2.1.6) apprennent directement la politique de l'expert et non, comme les approches d'ARI, une description succincte de la tâche sous la forme d'une fonction de récompense. L'imitation supervisée peut être vue comme moins problématique que l'ARI, disons certainement plus directe, puisque ce que ces méthodes cherchent à apprendre (la politique de l'expert) est de même nature que ce qui est présent dans les données. A l'inverse, pour les approches d'ARI il s'agit de comprendre la motivation à long terme de l'expert, et ce à partir d'exemples de décisions à portée immédiate. Il faut résoudre un problème de contexte temporel entre ce qui est démontré et ce que l'on cherche.

Les approches supervisées n'ont pas besoin pour fonctionner de données non expertes ou des probabilités de transition. Elles peuvent être appliquées lorsqu'on dispose uniquement de données expertes. Il est plus dur de fonctionner avec si peu de données lorsque l'on résout le problème de l'ARI, puisque l'objet recherché est plus complexe à trouver. Pourtant il existe, entre les outils développés dans le cadre de l'AR (section 2.2) et certaines approches de classification, des similitudes qui vont nous permettre de faire un parallèle entre les deux philosophies.

Dans le cas de la classification à fonction de score (Équation 2.14), le classifieur ap-

prend, à partir des données expertes, une fonction q qui permet de définir simplement la politique de classification π^C par une optimisation de ce score sur les actions (Équation 2.14) :

$$\forall s, \pi^C(s) \in \arg\max_{a \in \mathcal{A}} q(s,a). \tag{5.1}$$

Or la politique de l'expert, qui par hypothèse est une politique optimale, obéit à une relation du même type vis-à-vis de la fonction de qualité optimale pour la récompense inconnue R^E dont l'expert cherche à maximiser le cumul (Équation 2.48) :

$$\forall s \in \mathcal{S}, \pi^E(s) \in \arg\max_{a \in \mathcal{A}} Q^{\pi^E}_{R^E}(s,a). \tag{5.2}$$

La fonction de score q apprise par la plupart des approches d'imitation supervisée peut donc être vue comme un objet de même type qu'une fonction de qualité, dont nous avons dit en section 2.2 qu'elle fait le lien entre une optimisation myope, à l'échelle de la décision immédiate, et le critère à optimiser sur le long terme, le cumul des récompenses.

Il est problématique que l'apprentissage supervisé ne tient pas compte de cette structure temporelle dans la résolution du problème de l'imitation. Tel qu'il est posé, le problème de classification suppose que les attributs choisis encodent l'information nécessaire à la prise de décision pour chaque état or, comme tend à le montrer ce lien entre classification et AR, l'optimisation doit avoir lieu sur un critère à long terme qui prend la dynamique du système en compte.

Il est possible d'effectuer un travail d'ingénierie sur les attributs (sous-section 2.1.3) qui permette aux méthodes supervisées de bien généraliser, mais nous souhaitons ne pas trop avoir à faire appel à l'expertise humaine pour bâtir des systèmes intelligents. Il faut trouver un moyen d'informer les méthodes supervisées de la structure temporelle du problème, afin d'éviter le décalage actuel entre un problème d'optimisation mettant en jeu une fonction de score qui remplit le rôle d'une fonction de qualité, et qui pourtant provient de l'exploitation directe de données concernant les décisions immédiates de l'expert.

L'algorithme CSI du chapitre suivant et SCIRL, que nous présentons ici, font précisément cela. Nous exposons la logique derrière SCIRL dans la section 5.2, puis en présentons une justification théorique (section 5.3) et expérimentale (section 5.4).

5.2 *Description*

Nous décrivons les principe général de SCIRL sous-section 5.2.1, puis présentons une instanciation particulière [2] (sous-section 5.2.2) qui sera celle utilisée pour les expériences (section 5.4).

2. Qui grâce à une heuristique peut fonctionner en utilisant uniquement des données expertes.

5.2.1 *Principe général*

L'un des résultats principaux de la section 2.2 est la notion de politique gloutonne, qui en optimisant à l'échelle de la décision immédiate la fonction de qualité parvient à optimiser le critère long terme du cumul des récompenses. Les approches de classification à fonction de score souffrent de ce que la fonction q ne présente pas la même structure temporelle qu'une fonction de qualité bien qu'elle remplit un rôle similaire, comme nous venons de le découvrir.

Afin de permettre aux techniques de classification d'exploiter cette structure temporelle qui leur manque, nous allons étudier comment introduire l'attribut moyen dans la

fonction de score. Nous avons vu Équation 4.3 que l'attribut moyen est la généralisation vectorielle de la fonction de valeur, qui porte cette structure temporelle indispensable à la réconciliation de l'optimisation locale et long terme. De plus, de par sa définition, il est naturellement présent dans la fonction de qualité $Q^{\pi^E}_{R_\theta}$ pour la récompense paramétrée linéairement R_θ :

$$Q^{\pi^E}_{R_\theta}(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R_\theta(s_t,a_t)\middle| s_0=s, a_0=a, \pi\right] \tag{5.3}$$

$$= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t \theta^T \phi_R(s_t,a_t)\middle| s_0=s, a_0=a, \pi\right] \tag{5.4}$$

$$= \theta^T \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t \phi_R(s_t,a_t)\middle| s_0=s, a_0=a, \pi\right] \tag{5.5}$$

$$= \theta^T \mu^E(s,a). \tag{5.6}$$

Sachant tout cela, nous choisissons un classifieur à fonction de score q_θ qui puisse être paramétrée linéairement. Comme nous avons identifié le rôle de la fonction de score et celui de la fonction de qualité optimale, nous utilisons comme attribut pour la paramétrisation linéaire de la fonction de score une approximation $\hat{\mu^E}$ de l'attribut moyen de l'expert μ^E afin d'obtenir une expression identique à l'Équation 5.6 :

$$q_\theta = \theta^T \hat{\mu}^E. \tag{5.7}$$

L'algorithme LSTD-μ du chapitre 4 est tout indiqué pour fournir cette estimation.

Ce que le classifieur apprend en réalité quand il apprend la fonction de score est le vecteur de paramètres θ, qui est également le vecteur de paramètres de la fonction de récompense (Équation 2.53). Ainsi, en résolvant de manière supervisée le problème de l'imitation, on trouve une récompense (qui est l'objet d'intérêt de l'ARI), simplement en paramétrant la classification avec un objet porteur de la structure temporelle du PDM. Tout cela est résumé algorithme 10.

Algorithme 10: Principe générique de SCIRL

Entrées : Une base d'entraînement établie par l'expert $D^{\pi^E}_{sas}$;
Données : Une approximation $\hat{\mu}^E$ de l'attribut moyen de l'expert;
Un classifieur à fonction de score paramétrée linéairement;
Sorties : Un vecteur de paramètres θ pour la récompense $R_\theta = \theta^T \phi_R$

1 Paramétrer la fonction de score q_θ du classifieur comme cela :

$$q_\theta = \theta^T \hat{\mu}^E \tag{5.8}$$

Entraîner le classifieur sur la base de données $D^{\pi^E}_{sas}$;

2 **retourner** θ

5.2.2 *Instanciation avec heuristique*

Si la majorité des approches de classification peuvent être vue comme utilisant une fonction de score, celle-ci n'est pas forcément paramétrée linéairement d'une manière qui nous permette d'appliquer SCIRL. Fort heureusement, nous avons étudié un classifieur se prêtant bien à cet exercice en sous-section 2.1.3. L'apprentissage se réduit à une

descente de sous-gradient :

$$\nabla J(\theta) = \sum_{i=1}^{N} \left(\hat{\mu}^E(s_i, a_i^*) - \hat{\mu}^E(s_i, a_i) \right) + \lambda\theta, \tag{5.9}$$

$$\text{avec : } a_i^* = \arg\max_{a \in \mathcal{A}} \theta^T \hat{\mu}^E(s_i, a) + \mathfrak{l}(s_i, a), \tag{5.10}$$

et $\mathfrak{l}$ telle que définie Équation 2.18.

Cette méthode de classification est aussi celle employée par RATLIFF, BAGNELL et ZINKEVICH [RBZ06] dans leur algorithme MMP. Une différence importante entre SCIRL et MMP est que ce dernier prend la structure temporelle du problème en compte en associant des politiques à des PDM. Cela fonctionne mais implique de devoir résoudre ces PDM et fait donc de MMP une approche itérative conforme au schéma de l'algorithme 4. Ici ce sont les actions sont associées aux états, comme dans les approches supervisées (section 2.1). La structure temporelle est introduite par l'attribut moyen.

Le calcul du sous-gradient $\nabla J(\theta)$ implique de pouvoir évaluer l'attribut moyen de l'expert μ^E en tout état présent dans la base d'entraînement $D_{sas}^{\pi^E}$, pour toute action.

Si les données de l'expert sont organisées en trajectoires, sur une trajectoire de longueur L, il est possible d'estimer μ^E sur tous les couples (s_i, a_i) de la trajectoire grâce à une simple estimation de Monte-Carlo :

$$\hat{\mu}^E(s_i, a_i) = \sum_{t=0}^{L-1-i} \gamma^t \phi_R(s_{t+i}, a_{t+i}). \tag{5.11}$$

Il faut noter que la qualité de l'estimation risque de diminuer à mesure que l'on s'approche de la fin de la trajectoire, puisque le nombre d'échantillons participant au calcul diminue. Cette méthode fonctionne bien sur les problèmes quasi-déterministes mais risque de poser problème sur les dynamiques plus stochastiques (la variance sera grande car une seule trajectoire est utilisée). Utiliser LSTD-μ est la solution la plus souple. Si seules les données de l'expert sont disponibles, une version *on-policy* de l'algorithme permettra d'obtenir une estimation $\hat{\mu}^E$ sur les couples (s_i, a_i) présents dans la base d'entraînement. Si des données non expertes sont également disponibles, alors LSTD-μ dans sa version *off-policy* saura les exploiter (si l'on peut interroger l'expert sur un état arbitraire).

Le calcul de ${a_i}^*$ dans le sous-gradient implique, pour un état de la base d'entraînement, de connaître l'attribut moyen de l'expert pour toute action, et non juste celle choisie par l'expert. En l'absence de donnée, il est impossible de pallier ce problème d'une manière qui soit formellement juste. Nous pouvons en revanche employer une heuristique dont l'étude expérimentale (section 5.4) montrera la validité.

Pour introduire intuitivement l'heuristique, imaginons dans une voiture à double commande telles celles employées par les auto-écoles un expert au volant, la tâche étant de conduire. Imaginons maintenant qu'un plaisantin dans le siège passager applique un instant une force sur le volant, ou appuie brièvement et légèrement sur une des pédales. On voit qu'à l'échelle de la tâche cette perturbation est minime et le conducteur expert a tôt fait de poursuivre tel qu'il l'aurait fait si la perturbation n'avait pas eu lieu. De la même manière, on suppose qu'une perturbation causée par une action parasite lorsqu'on doit évaluer $\mu^E(s, a \neq \pi^E(s))$ est immédiatement absorbée par une action correctrice de l'expert, et donc que la seule conséquence de la perturbation est de faire perdre un pas de temps à l'expert. Ce qui se traduit mathématiquement par :

$$\hat{\mu}^E(s_i, a \neq a) = \gamma \hat{\mu}^E(s_i, a_i) \tag{5.12}$$

Armés de notre méthode de classification à fonction de score linéairement paramétrée ainsi que de notre heuristique pour le calcul de l'attribut moyen de l'expert, il est maintenant possible d'instancier SCIRL afin qu'il apprenne une récompense uniquement nourri de données expertes (algorithme 11).

Algorithme 11: SCIRL uniquement sur des données expertes

Entrées : Une base d'entraînement établie par l'expert $D_{sas}^{\pi^E}$;

Données : Une approximation $\hat{\mu}^E$ de l'attribut moyen sur tous les états présents dans la base $D_{sas}^{\pi^E}$;

L'extension de cette approximation à toutes les actions grâce à l'heuristique de l'Équation 5.12;

Sorties : Un vecteur de paramètres θ pour la récompense $R_\theta = \theta^T \phi_R$

1 Initialiser θ arbitrairement;

2 Procéder à la descente de sous-gradient dont le sous-gradient est :

$$\nabla J(\theta) = \sum_{i=1}^{N} \hat{\mu}^E \left(s_i, \arg\max_{a \in \mathcal{A}} \theta^T \hat{\mu}^E(s_i, a) + \mathbb{1}(s_i, a) \right) - \hat{\mu}^E(s_i, a_i) + \lambda\theta$$

retourner θ

5.3 *Validation théorique*

Maintenant que le principe de SCIRL ainsi qu'une instanciation particulière utilisant l'algorithme 1 et LSTD-μ sont en place, Nous fournissons des garanties théoriques sur les performances d'imitation de SCIRL [3].

3. Chronologiquement postérieures à celles concernant l'algorithme CSI, qui sont dues au travail d'un autre doctorant de l'équipe MaLIS, Bilal Piot.

Lorsque nous avons posé le problème de l'ARI, nous avons discuté des critères permettant d'évaluer la qualité de la solution. Dans l'analyse théorique de SCIRL, c'est le critère de l'Équation 2.52 qui nous intéresse. Il mesure l'optimalité de la politique de l'expert pour la récompense renvoyée par SCIRL. Nous allons le borner par un terme qui dépend d'objets dont certains n'ont pas encore été introduits.

Le premier de ces objet est le coefficient de concentration C_f [Muno7], qui mesure l'écart entre les distributions induites par toutes les politiques. Ici il s'agit d'un maximum qui sélectionne la séquence de politiques dont la distribution est la plus "différente" possible de celle de l'expert :

$$C_f = (1-\gamma) \sum_{t \geq 0} \gamma^t c(t) \tag{5.13}$$

$$\text{avec } c(t) = \max_{\pi_1, \ldots, \pi_t, s \in \mathcal{S}} \frac{({\rho_E}^\top P^{\pi_1} \ldots P^{\pi_t})(s)}{\rho_E(s)}. \tag{5.14}$$

Nous rappelons également la définition de l'erreur de classification $\epsilon_C^{\rho_E}$ (Équation 2.5) :

$$\epsilon_C^{\rho_E} = \mathbb{E}\left[\mathbb{1}(\pi^C(s) \neq \pi^E(s)) \middle| s \sim \rho_E \right]. \tag{5.15}$$

L'erreur d'estimation de l'attribut moyen de l'expert est quantifiée par :

$$\epsilon_\mu = \hat{\mu^E} - \mu^E \in (\mathbb{R}^{d_{\phi_R}})^{\mathcal{S} \times \mathcal{A}}. \tag{5.16}$$

En conséquence, L'erreur entre la fonction de score apprise par le classifieur q_θ et la

fonction de qualité de l'expert $Q_{R_\theta}^{\pi^E}$ pour la récompense de SCIRL vaut :

$$\epsilon_Q = q_\theta - Q_{R_\theta}^{\pi^E} \tag{5.17}$$
$$= \theta^T(\hat{\mu^E} - \mu^E) \tag{5.18}$$
$$= \theta^T \epsilon_\mu \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}. \tag{5.19}$$

C'est l'espérance sur la distribution stationnaire de l'expert de l'écart maximum entre les actions de ce dernier terme qui intervient dans la borne :

$$\bar{\epsilon}_Q = \mathbb{E}\left[\max_{a\in\mathcal{A}} \epsilon_Q(s,a) - \min_{a\in\mathcal{A}} \epsilon_Q(s,a) \middle| s \sim \rho_E\right]. \tag{5.20}$$

Nous avons maintenant introduit les notions nécessaires à l'expression de la borne théorique sur la qualité de la récompense fournie par SCIRL :

Théorème 1 (Borne des performances de SCIRL)**.** *Soit R_θ la fonction de récompense renvoyée par l'algorithme 10. Soient C_f, $\epsilon_C^{\rho_E}$ et $\bar{\epsilon}_Q$ les quantités définies ci-dessus. On a :*

$$0 \leq \mathbb{E}\left[V_{R_\theta}^{\pi^*_{R_\theta}} - V_{R_\theta}^{\pi^E} \middle| s \sim \rho_E\right] \leq \frac{C_f}{1-\gamma}\left(\bar{\epsilon}_Q + \epsilon_C^{\rho_E}\frac{2\gamma\|R_\theta\|_\infty}{1-\gamma}\right). \tag{5.21}$$

La preuve est reportée en annexe A.

Le terme $\|R_\theta\|_\infty$ (Le choix de la norme est d'une importance limitée) n'est pas problématique car nous savons depuis la section 3.1 qu'il est possible de normaliser θ sans perte de généralité.

Le terme le moins contrôlable de cette borne est l'erreur $\bar{\epsilon}_Q$, qui dépend de la qualité de l'estimation de l'attribut moyen $\hat{\mu^E}$. L'usage de LSTD-μ permet cependant de profiter des garanties offertes section 4.2.

Lorsque les deux termes d'erreur sont nuls, nous découvrons un corollaire intéressant :

Corollaire 1 (SCIRL dans le cas idéal)**.** *Si $\epsilon_C^{\rho_E} = 0$ et $\bar{\epsilon}_Q = 0$, alors π^E est l'unique politique optimale pour la récompense R_θ renvoyée par SCIRL.*

Démonstration. Soit π^H une politique optimale pour R_θ. Comme π^H et π^E sont toutes deux optimales, alors d'après l'Équation 2.47 :

$$\pi^H \in g\left(\pi^E\right) \tag{5.22}$$
$$\Rightarrow \forall s \in \mathcal{S}, \pi^H(s) = \arg\max_{a\in\mathcal{A}} Q_{R_\theta}^{\pi^E}(s,a) \tag{5.23}$$
$$= \arg\max_{a\in\mathcal{A}} \theta^T \mu^E(s,a). \tag{5.24}$$

Comme $\bar{\epsilon}_Q = 0$,

$$\forall s \in \mathcal{S}, \pi^H(s) = \arg\max_{a\in\mathcal{A}} q_\theta(s,a) \tag{5.25}$$
$$= \pi^C(s). \tag{5.26}$$

Et comme $\epsilon_C^{\rho_E} = 0$,

$$\forall s \in \mathcal{S}, \pi^H(s) = \pi^E(s). \tag{5.27}$$

□

Une autre propriété de SCIRL est d'éviter les solutions dégénérées, en effet la solution dégénérée par excellence, la récompense nulle, minimise le critère utilisé dans la borne[4] mais implique que la politique de classification π^C est la politique aléatoire uniforme, atteignant l'erreur de classification maximale $\epsilon_C^{\rho_E} = \frac{|\mathcal{A}|-1}{|\mathcal{A}|}$. Le classifieur cherchant à minimiser $\epsilon_C^{\rho_E}$, il est peu probable que ce cas advienne.

4. Pour $\theta = 0$, π^E est optimale pour R_θ, de même que n'importe quelle autre politique.

5.4 *Validation empirique*

Si l'agent est aux commandes d'un système où la moindre déviation par rapport au contrôle optimal entraîne un échec irrécupérable [5], alors l'heuristique proposée Équation 5.12 ne permet probablement pas une bonne approximation de l'attribut moyen de l'expert, puisqu'elle est intuitivement basée sur le principe que l'expert est capable de revenir sur la bonne trajectoire après qu'un mauvais choix d'action a été effectué. Il faut donc que l'opérateur, en bâtissant la représentation du problème, soit vigilant à ce phénomène. Nous illustrons empiriquement dans cette section que les propriétés nécessaires au bon fonctionnement de SCIRL ne sont pas difficiles à réunir.

5. Par échec irrécupérable, il est entendu une déviation vers une partie de l'espace d'état qui ne communique pas ou peu avec le reste, et de fait garde l'agent prisonnier sans possibilité de sortie. L'existence d'une telle zone dont la sortie serait rigoureusement impossible violerait la condition d'ergodicité dont nous avons parlé sous-section 2.2.1.

5.4.1 *Introduction*

Contrairement à la section précédente, qui étudie le critère de l'Équation 2.52 quantifiant le degré d'optimalité de la politique experte vis-à-vis de la récompense trouvée par SCIRL, nous utiliserons ici un oracle pouvant observer des critères semblables à celui de l'Équation 2.51, qui mesure l'optimalité de la politique optimale pour la récompense trouvée par SCIRL, par rapport à la récompense inconnue optimisée par l'expert. Nous profitons en cela de ce que la récompense optimisée par l'expert nous est connue sur ces problèmes jouet.

La politique dont l'optimalité est mesurée est issue de l'optimisation d'un PDM, puisqu'elle est optimale pour la récompense R_θ trouvée par SCIRL. Il faut bien comprendre que cette étape de résolution d'un PDM ne fait pas partie de SCIRL et est indépendante de la résolution du problème de l'ARI. L'objet d'intérêt de l'ARI est la récompense R_θ, nous ne l'optimisons que dans un but d'évaluation, par un procédé dont nous préciserons la nature dans la description des expériences, mais qui n'a pas de réel intérêt. Dans une approche pratique d'imitation par le biais de l'ARI, il faudrait bien entendu tenir compte de la contrainte qu'est la nécessité de résoudre le PDM une fois la récompense trouvée. Il existe de multiples manières de faire [6], le choix de la technique utilisée dépend énormément du contexte et rejoint le problème vaste de l'apprentissage du contrôle optimal. Nous choisissons ici d'exposer des approches efficaces de l'ARI, facilement applicables à de multiples domaines, sans nous soucier de savoir comment la récompense sera optimisée, tant cette dernière étape est dépendante du problème. Nous la voyons comme un problème parallèle à celui de l'ARI.

6. En l'absence de simulateur, il faudra avoir recours à des méthodes *batch* et *off-policy*.

Nous testons SCIRL dans l'instanciation proposée algorithme 11, en utilisant l'Équation 5.11 pour l'estimation de l'attribut moyen de l'expert sur deux problèmes jouet, le *mountain-car* sous-section 5.4.2 et le *highway* sous-section 5.4.3. Dans les deux cas, nous le comparons d'une part à une approche d'imitation supervisée et d'autre part à l'algorithme d'ARI existant à notre connaissance le mieux adapté à un scénario où la collecte de données est coûteuse, RelEnt, dont la version échantillonnée que nous utilisons est décrite algorithme 5. La représentation choisie (attributs, sous-section 2.1.3, ou noyau, sous-section 2.1.5) est toujours une représentation naturelle et simple à mettre en œuvre, ne présentant aucune caractéristique *ad-hoc*.

Rappelons que la capacité de généralisation des méthodes supervisées dépendant en grande partie du soin apporté à la définition de cette représentation, ces expériences devraient montrer la supériorité des méthodes d'ARI face aux méthodes supervisées en ce qui concerne la généralisation à des parties de l'espace d'état n'ayant pas été explorées par l'expert. Grâce à l'extraction de la fonction de récompense, description succincte de la tâche à effectuer, les algorithmes d'ARI sont capables de guider l'agent

lorsqu'il apprend le contrôle optimal dans la zone inexplorée, tandis que les méthodes supervisées restent myopes et ne peuvent utiliser toute l'information contenue dans les échantillons de l'expert.

Ces deux expériences devraient également montrer la validité de l'heuristique qui permet à SCIRL de fonctionner avec pour seule information les échantillons provenant de l'expert. Il faut que les performances de SCIRL soient au moins aussi bonnes que celle de RelEnt, qui lui dispose d'échantillons aléatoires lui donnant des informations sur la dynamique du système (et ne peut utiliser notre heuristique).

5.4.2 *SCIRL appliqué au* mountain-car

Comme nous l'avons précisé section B.2, la démonstration de l'expert ignore toute une partie de l'espace d'état, le quadrant Sud-Est où la voiture est proche du but, mais s'en éloigne. Les données de l'expert sont collectées, en choisissant un état de départ telle que la position est dans l'intervalle $[-1.2; -0.9]$ et la vitesse dans $[-0.07; 0]$. La trajectoire est poursuivie jusqu'à ce que l'expert atteigne le but. Autant de trajectoires que nécessaire sont effectuées pour obtenir des ensembles de 10, 30, 100 et 300 échantillons. La dernière trajectoire est tronquée pour atteindre exactement le nombre d'échantillons voulu (cela signifie typiquement que pour les ensemble de 10 et 30 échantillons, l'expert ne fournit même pas une trajectoire complète).

L'algorithme RelEnt ne fonctionnant pas avec uniquement des données de l'expert, un base d'entraînement supplémentaire $D_{sas}^{\sim}$ lui sera fournie. Cette base est constituée de 1000 trajectoires de longueur 5, débutant dans un état choisi au hasard uniformément dans tout l'espace d'état, les actions étant choisies également au hasard. Il est intéressant de noter que cette base contient d'un à deux ordres de grandeur plus d'échantillons que la base experte.

Cette même base $D_{sas}^{\sim}$ sera utilisée par LSPI pour optimiser les récompenses trouvées par SCIRL et RelEnt.

Les attributs utilisés par LSPI, SCIRL et RelEnt sont les mêmes, il s'agit d'un maillage de gaussiennes tel que décrit Équation 2.12 et illustré Figure 2.1.

Le classifieur dont les performances sont illustrées est une SVM à noyau gaussien dont les hyper-paramètres sont choisis automatiquement par l'implémentation utilisée[7]. Les performances de cette modèle étant très légèrement supérieures à celles du classifieur structuré de Taskar, Chatalbashev, Koller et Guestrin [Tas+05] (algorithme 1), ce sont celles que nous affichons ici.

7. Module SVM de Scikit-learn

En ordonnée sur la Figure 5.1, nous représentons le nombre de pas nécessaire à l'agent pour qu'il atteigne le but (plafonné à 300), lorsque l'état de départ est tiré aléatoirement dans l'espace d'état entier. Les valeurs affichées sont la moyenne et l'écart-type sur 100 expériences. Ce critère est directement lié à la valeur de la politique pour la récompense inconnue R^E : si l'expert touche une récompense de 1 lorsqu'il atteint l'objectif, et que la récompense est nulle partout ailleurs, du fait de l'amortissement temporel la valeur de sa politique en un état vaut γ^T où T est le nombre de pas qui le sépare de l'objectif. En conséquence, ce que nous traçons est le logarithme en base γ de la valeur moyenne de la politique.

Les résultats présentés Figure 5.1 sont conformes à ce qui était espéré, en ce sens que les deux algorithmes d'ARI sont plus performants que la méthode supervisée. On constate cependant que les performances de RelEnt sont beaucoup plus proches de celles de la classification qu'elles ne le sont de celles de SCIRL. Ce dernier parvient dès 30 échantillons à des performances meilleures que celles des deux autres méthodes

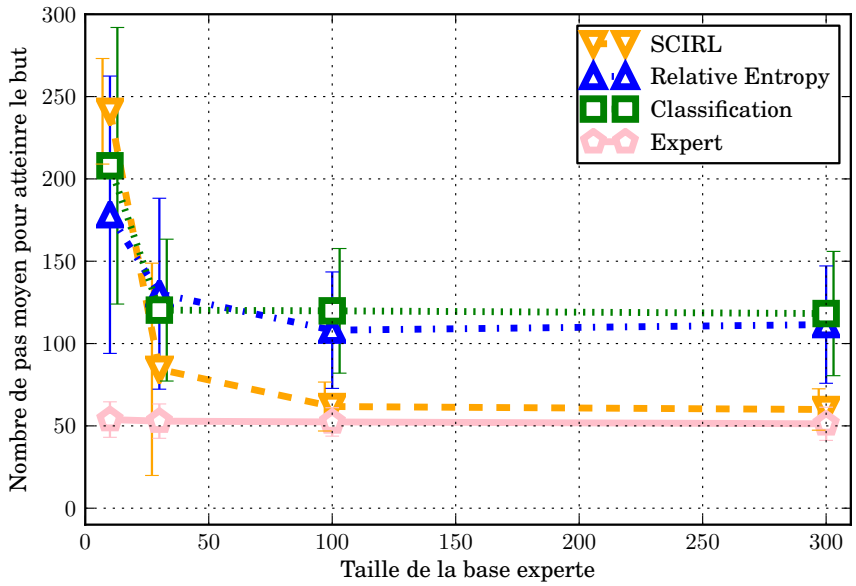


FIGURE 5.1: Algorithmes d'imitation sur le *mountain-car*. La politique de l'expert est également évaluée afin d'offrir un point de référence.

quelle que soit la quantité de données expertes. Quand au moins une trajectoire complète est présente dans les données, la politique issue de la récompense trouvée par SCIRL est presque aussi bonne que celle de l'expert.

Le calcul de l'attribut moyen de l'expert pour SCIRL est ici effectué avec l'Équation 5.11 car le problème est déterministe.

En utilisant un peu plus[8] d'information que la méthode de classification, et moins que l'autre méthode d'ARI, SCIRL est en mesure de trouver une récompense menant à une imitation de qualité.

8. La classification utilise une base $D_{sa}^{\pi^E}$ tandis que SCIRL utilise une base $D_{sas}^{\pi^E}$ où il convient d'enregistrer l'état d'arrivée.

5.4.3 *SCIRL appliqué au* highway

Le problème du simulateur de conduite sur autoroute est défini section B.3. L'espace est discret, mais grand. Les attributs choisis sont les attributs standards décrits Équation 2.9. La méthode supervisée de classification structurée considérée cette fois ci (algorithme 1) utilise ces attributs, et ne va donc pas disposer d'information particulière sur la dynamique ou la sémantique du problème. Les démonstrations de l'expert étant fort courtes, il va falloir attendre que celui-ci explore une vaste partie de l'espace d'état avant que la classification ne parvienne à apprendre un contrôle correct. L'expert fournit une base de données $D_{sas}^{\pi^E}$ organisée sous la forme de N trajectoires de longueur N, avec N prenant successivement pour valeur 3, 7, 10, 15 et 20.

L'expert est un agent optimal entraîné sur une récompense pénalisant les collisions (-1 par collision) et les sorties de route (-0.5 par sortie de route) mais exigeant une grande vitesse (1 si la vitesse est maximale).

RelEnt nécessite pour fonctionner une autre base de données $D_{sas}^{\sim}$. Celle-ci est constituée de 100 marches aléatoires de longueur 10, démarrant dans un état choisi aléatoirement. Il y a cette fois-ci encore besoin d'un nombre d'échantillons d'un à deux ordres de grandeur plus grand que les démonstrations expertes.

SCIRL, pour calculer l'attribut moyen de l'expert, utilise la méthode de l'Équation 5.11

car l'inversion de matrice nécessaire au fonctionnement de LSTD-μ est, lorsque répétée sur les 100 expériences que nous avons menées pour obtenir l'information de variance, un petit peu longue à effectuer. Il serait possible de régler ce problème en choisissant les attributs avec plus de soin, mais nous souhaitons montrer que SCIRL est une méthode générique qui ne nécessite que peu ou pas de travail de la part de l'opérateur.

L'espace d'état étant discret et la dynamique connue, nous pouvons optimiser les récompenses trouvées par les algorithmes d'ARI avec une méthode de PD. Le critère que nous utilisons pour comparer les différentes politiques est la moyenne sur tous les états de la valeur de la politique. Même les démonstrations les plus longues de l'expert (400 échantillons) n'explorent qu'une petite partie de l'espace d'état-action (de cardinal 3645), il est donc primordial de bien généraliser pour obtenir un contrôle correct.

Les résultats obtenus sont donnés Figure 5.2. Les résultats présentent bien la hiérarchie à laquelle nous nous attendions : l'algorithme supervisé ne parvient à apprendre un bon contrôle que petit à petit, à mesure que la taille de la base de données experte croît. L'ordre est le même que pour le problème du *mountain-car*, mais les performances de RelEnt diffèrent en ceci qu'elles sont maintenant beaucoup plus proches de celles de SCIRL (qui reste malgré tout au dessus) que de la classification.

Dans un domaine où les attributs (discrets) n'ont pas été en mesure de fournir à la méthode supervisée la moindre information *a priori* sur le système (précédemment pour le *mountain-car*, le noyau gaussien de la SVM possède la propriété de cohérence spatiale [9]), les méthodes d'ARI présentent de bien meilleures capacités de généralisation.

9. Deux états proches auront des attributs proches.

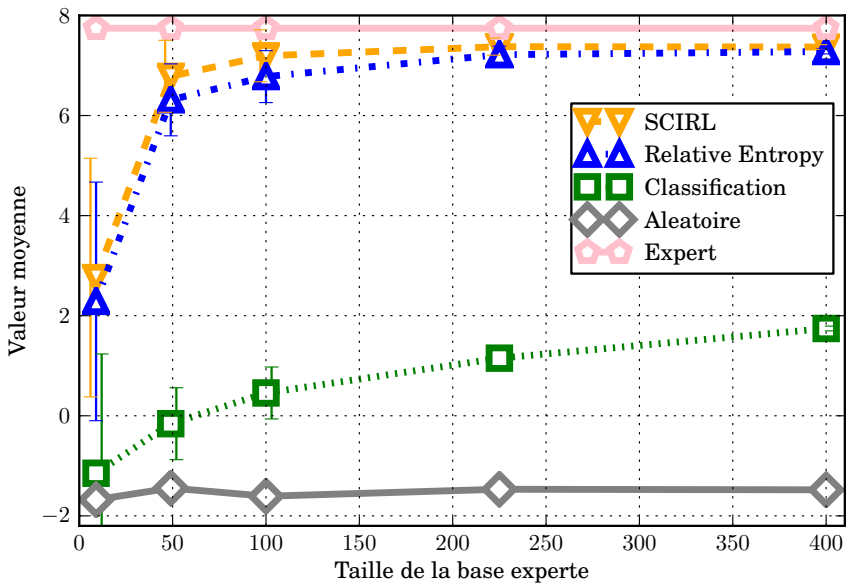


FIGURE 5.2: Algorithmes d'imitation sur le *highway*. En plus de la valeur de l'expert comme dans la Figure 5.1, la valeur d'une politique apprise sur une récompense aléatoire donne un point de référence de valeur minimale. On montre ici l'écart-type et la moyenne sur 100 expériences.

5.5 *Conclusion*

Dans les deux expériences que nous venons de décrire, les méthodes supervisées ne peuvent qu'imiter l'expert là où sa démonstration donne des informations. La capacité

de généralisation est sévèrement limitée par l'absence de savoir *a priori* dans la définition du problème. Les algorithmes d'ARI, à l'inverse, sont capables, à partir d'une démonstration locale, d'extraire une description satisfaisante de la tâche accomplie par l'expert.

SCIRL en particulier a pu fonctionner à partir des mêmes données que la méthode supervisée (à l'ajout de l'état d'arrivée s' près), sans données supplémentaires, à l'inverse de RelEnt. Il a su exploiter les informations contenues dans les transitions pour introduire la dynamique du système dans la résolution du problème de l'imitation. L'heuristique que nous avons présentée Équation 5.12 et que la théorie ne permet pas facilement d'analyser a fait ses preuves sur des problèmes déterministes. SCIRL est moins gourmand en temps de calcul et en données qu'un algorithme d'AR comme par exemple LSPI. Il faut tenir compte de cela, puisque contrairement aux méthodes supervisées qui renvoient directement une politique de contrôle, SCIRL renvoie une récompense. Cette récompense devra par la suite être optimisée, en utilisant par exemple un algorithme d'AR auquel il faut fournir de l'information dont SCIRL n'a pas besoin. Le gain en performance justifie probablement l'effort lié à la résolution du PDM, des méthodes *ad-hoc* permettant de traiter les problèmes qui ne manqueront pas de se poser.

Les résultats théoriques et pratiques montrent que SCIRL résout bien le problème de l'ARI tel que nous l'avion posé en sous-section 2.3.1. Les soucis liés au choix des attributs, bien que toujours présents, sont moins préoccupants que pour les approches supervisées grâce à l'utilisation de l'attribut moyen, qui prend la dynamique en compte. L'estimation de celui-ci lorsqu'uniquement des données expertes sont disponibles est rendue possible grâce à l'usage d'une heuristique qui semble bien fonctionner en pratique. Les solutions dégénérées sont évitées grâce à la structure de SCIRL, qui adapte une méthode de classification. L'algorithme de l'état de l'art le plus à même de suivre SCIRL sur les problèmes où peu d'information est disponible, RelEnt, est moins performant et nécessite plus d'information. Si cependant cette information est facilement disponible, SCIRL peut en profiter grâce à l'emploi de LSTD-μ.

6

Apprentissage par renforcement inverse en cascadant classification et régression

Les avantages présentés par SCIRL découlent de l'identification de la fonction de score d'un classifieur à la fonction de qualité de l'expert. Cela permet d'introduire la dynamique du système dans une approche supervisée, combinant les avantages en terme de généralisation caractéristiques des approches d'ARI et la simplicité d'utilisation des approches supervisées.

Cependant SCIRL nécessite l'emploi d'un classifieur à fonction de score linéairement paramétrée, et l'estimation de l'attribut moyen de l'expert. Il existe comme nous allons le voir dans cette section un autre moyen de profiter de la similitude entre fonction de score et fonction de qualité. Nous présentons section 6.1 l'algorithme CSI, pour lequel il existe également une garantie théorique (section 6.2) ainsi qu'une étude empirique permettant de comparer ses performances à celles de SCIRL (section 6.3) et de valider l'heuristique qui, là encore, permet de dépasser le cadre de l'analyse théorique en n'utilisant que des données expertes.

6.1 Description

Notre nouvelle approche de l'ARI, CSI, est basée à l'instar de SCIRL sur le constat (section 5.1) de similitude entre la fonction de score q d'un classifieur et la fonction de qualité $Q^{\pi^E}_{R^E}$ de l'expert (qui par hypothèse est un agent optimal) pour la récompense inconnue R^E que justement, nous aimerions connaître.

6.1.1 Principe général

Là où cette approche diverge de SCIRL, c'est dans la manière d'introduire la dynamique du PDM en utilisant la fonction de score q apprise par un classifieur. Nous n'allons pas ici utiliser une paramétrisation astucieuse de cette fonction de score. Nous allons utiliser l'équation de Bellman (Équation 2.43) qui définit de manière récursive la fonction de qualité, et isoler la récompense. On rappelle que $\pi^E(s) \in \arg\max_{a\in\mathcal{A}} Q^{\pi^E}_{R^E}(s,a)$ car π^E est optimale. La fonction de qualité de l'expert [1] obéit donc alors à l'équation d'optimalité de Bellman :

1. Qui est donc une fonction de qualité optimale pour R^E

$$Q^{\pi^E}_{R^E}(s,a) = R^E(s,a) + \gamma \sum_{s'\in\mathcal{S}} p\left(s'|s,a\right) Q^{\pi^E}_{R^E}(s', \pi^E(s')), \tag{6.1}$$

donc la récompense R^E que l'on cherche suit :

$$R^E(s,a) = Q^{\pi^E}_{R^E}(s,a) - \gamma \sum_{s'\in\mathcal{S}} p\left(s'|s,a\right) Q^{\pi^E}_{R^E}(s', \pi^E(s')). \tag{6.2}$$

La fonction de score q d'un classifieur joue un rôle similaire à celui d'une fonction de qualité optimale :

$$\pi^C(s) \in \arg\max_{a \in \mathcal{A}} q(s,a) \tag{6.3}$$

En remplaçant la fonction de qualité de l'expert par son *alter-ego* la fonction de score d'un classifieur imitant l'expert, nous définissons la "récompense du classifieur" R^C [2] :

$$R^C(s,a) = q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p\left(s'|s,a\right) q(s', \max_{a'} q(s',a')), \tag{6.4}$$

$$= q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p\left(s'|s,a\right) q(s', \pi^C(s')). \tag{6.5}$$

2. La politique du classifieur π^C est une politique optimale pour la récompense R^C puisque du fait de l'Équation 6.3 et de la définition de R^C, q vérifie l'équation d'optimalité de Bellman pour R^C.

La présence dans cette équation des probabilités de transition ne nous permet pas d'espérer évaluer directement R^C, nous souhaitons en effet que CSI fonctionne dans un cadre où les données sont difficiles à réunir.

Supposons la disponibilité d'une base d'échantillons non experte $D^{\sim}_{sas}$:

$$D^{\sim}_{sas} = \{s_i, a_i, s'_i\}_{0 \leq i \leq N}. \tag{6.6}$$

L'état d'arrivée s'_i est tiré selon la distribution $p\left(\cdot|s_i,a_i\right)$, dès lors il est possible d'établir une version échantillonnée de l'Équation 6.5 :

$$\hat{r}_i = q(s_i,a_i) - \gamma q(s'_i, \pi^C(s'_i)). \tag{6.7}$$

L'idée générale de CSI et qui lui donne le nom de *cascadée*, est, après avoir appris la fonction de score q et la politique de classification π^C en entraînant un classifieur sur une base de données experte $D^{\pi^E}_{sa}$, de fournir à un régresseur la base de données

$$D^q_{sar} = \{\left(s_i,a_i\right), \hat{r}_i | s_i, a_i, s'_i \in D^{\sim}_{sas}\}, \tag{6.8}$$

afin qu'il apprenne $\hat{R}^C$, une approximation de la version échantillonnée de la récompense induite par la fonction de score du classifieur R^C. L'algorithme 12 résume cela.

Algorithme 12: Principe générique de CSI

Entrées : Une base d'entraînement établie par l'expert $D^{\pi^E}_{sa}$;
Une base d'entraînement non experte $D^{\sim}_{sas}$;
Données : Une méthode de classification à fonction de score;
Une méthode de régression;
Sorties : Une fonction de récompense $\hat{R}^C$

1 Entraîner le classifieur sur la base de données experte $D^{\pi^E}_{sa}$, obtenant ainsi q et π^C;
2 Établir la base d'entraînement

$$D^q_{sar} = \{\left(s_i,a_i\right), \hat{r}_i = q(s_i,a_i) - \gamma q(s'_i, \pi^C(s'_i)) | s_i, a_i, s'_i \in D^{\sim}_{sas}\};$$

Entraîner le régresseur sur la base D^q_{sar}, obtenant ainsi $\hat{R}^C$;
3 **retourner** $\hat{R}^C$

Grâce à l'utilisation de l'équation de Bellman, cascader deux approches supervisées permet (comme nous allons le montrer dans les deux sections qui suivent) de trouver une récompense. La plupart des méthodes de classification utilisent une fonction de score. N'importe quelle méthode de régression peut être utilisée pour la seconde étape. Cette grande latitude dans le choix d'un algorithme ou d'un autre pour chacune des étapes est une force de CSI qui peut ainsi profiter du large éventail de techniques de

l'apprentissage supervisé afin de limiter l'effort de l'opérateur lorsque cela est possible, ou de permettre à celui-ci d'adapter chaque étape au problème courant lorsque cela est nécessaire.

6.1.2 *Heuristique pour étendre CSI au cas où seules les données expertes sont disponibles*

La version générique de CSI repose sur la disponibilité de deux bases de données, une experte et l'autre non, comme c'est aussi le cas pour par exemple RelEnt. Nous avons vu que SCIRL peut, grâce à une heuristique, fonctionner à l'aide d'uniquement des données expertes. Il est souhaitable d'étendre cette capacité à CSI. Voyons comment l'absence de base d'entraînement $D_{sas}^{\sim}$ impacte l'algorithme, et voyons surtout comment compenser ce manque.

Si l'on ne dispose que d'une base d'entraînement experte $D_{sas}^{\pi^E}$ et que l'on essaye de bâtir la base D_{sar}^{q} avec, nous ne disposerons que d'échantillons :

$$\left(s_i, \pi^E(s_i)\right), \hat{r}_i = q(s_i, \pi^E(s_i)) - \gamma q(s'_i, \pi^C(s'_i)). \tag{6.9}$$

Ces données, les seules dont on dispose, sont tirées selon la dynamique induite par la politique experte. Or un PDM est plus riche que la seule chaîne de Markov induite par une politique. Particulièrement, il n'y a parmi ces échantillons aucune information concernant les couples $(s_i, a \neq \pi^E(s_i))$. Entraîner un régresseur sur ces seules données revient à espérer que le régresseur parviendra à généraliser à des actions qu'il n'a jamais vues. À moins de disposer d'information *a priori* et de parvenir à les intégrer au régresseur, cela est probablement une mauvaise idée. Typiquement, rien n'empêche le régresseur de décider que les couples $(s_i, a \neq \pi^E(s_i))$ sont associés à une valeur plus grande que $\hat{r}_i$.

Pour contraindre les solutions au problème de l'ARI, dont on a vu qu'il en existait un nombre infini dont certaines dégénérées, beaucoup d'algorithmes existants ([AN04 ; SBS08 ; RSB09 ; RA07]) prennent le parti d'affirmer que le choix d'une action effectué par l'expert doit être meilleur (selon un certain critère) que les autre choix possibles, par une certaine marge. Notre heuristique adopte le même parti pris en injectant dans la base D_{sar}^{q} les échantillons :

$$\left(s_i, \forall a \neq \pi^E(s_i)\right), \hat{r}_{min} = \min_{i \in [\![1;N]\!]} \hat{r}_i - 1. \tag{6.10}$$

Cela signifie en effet que la fonction apprise par le régresseur ne doit pas associer à un choix en désaccord avec une démonstration de l'expert une récompense supérieure ou égale à n'importe quel autre choix expert.

La récapitulation de l'approche CSI en utilisant cette heuristique est fournie algorithme 13.

6.2 *Validation théorique*

La borne que nous présentons quant aux performances de CSI est comme celle de SCIRL basée sur le critère de l'Équation 2.52. Elle montre qu'en contrôlant les erreurs de chacune des deux étapes supervisées, la récompense fournie par CSI devient telle que l'expert est optimal vis-à-vis d'elle.

Pour exprimer cette borne, nous avons besoin de l'erreur de classification, dont la définition est toujours la même :

$$\epsilon_C^{\rho_E} = \mathbb{E}\left[\mathbb{1}(\pi^C(s) \neq \pi^E(s)) \middle| s \sim \rho_E\right]. \tag{6.11}$$

Algorithme 13: CSI dans sa version utilisant l'heuristique

Entrées : Une base d'entraînement établie par l'expert $D_{sas}^{\pi^E}$;
Données : Une méthode de classification à fonction de score;
Une méthode de régression;
Sorties : Une fonction de récompense $\hat{R}^C$

1 Entraîner le classifieur sur la base de données experte $D_{sas}^{\pi^E}$, obtenant ainsi q et π^C;
2 Établir la base d'entraînement

$$D_{sar}^{q} = \left\{ \left(s_i, a_i = \pi^E(s_i) \right), \hat{r}_i = q(s_i, \pi^E(s_i)) - \gamma q(s_i', \pi^C(s_i')) | s_i, a_i, s_i' \in D_{sas}^{\pi^E} \right\}$$
$$\cup \left\{ \left(s_i, \forall a \neq \pi^E(s_i) \right), \hat{r}_{min} = \min_{i \in [\![1;N]\!]} \hat{r}_i - 1 \right\};$$

Entraîner le régresseur sur la base D_{sar}^{q}, obtenant ainsi $\hat{R}^C$;
3 **retourner** $\hat{R}^C$

Nous utiliserons également l'erreur ϵ_R due au régresseur :

$$\epsilon_R = \max_{\pi \in A^S} \|\epsilon_\pi^R\|_{1,\rho_E} \tag{6.12}$$
$$\text{avec : } \epsilon_\pi^R(\cdot) = R^C(\cdot, \pi(\cdot)) - \hat{R}^C(\cdot, \pi(\cdot)), \tag{6.13}$$
$$\text{et la norme } \ell_1 \text{ } \rho\text{-pondérée : } \|f\|_{1,\rho} = \mathbb{E}\left[|f(x)| \,|\, x \sim \rho \right], \tag{6.14}$$

ainsi que l'écart de score :

$$\Delta q = \max_{s \in \mathcal{S}} (\max_{a \in A} q(s,a) - \min_{a \in A} q(s,a)) = \max_{s \in S} (q(s, \pi^C(s)) - \min_{a \in A} q(s,a)), \tag{6.15}$$

que nous pouvons normaliser, puisque le choix de l'action se fait sur les valeurs comparées et non absolues des scores.

Enfin, nous introduisons un coefficient de concentration plus fin que celui que nous avons utilisé pour borner la performance de SCIRL, qui montre le désaccord de distributions induite par une politique π et la politique de l'expert :

$$C_\pi = (1-\gamma) \sum_{t \geq 0} \gamma^t c_\pi(t) \tag{6.16}$$
$$\text{avec : } c_\pi(t) = \max_{s \in S} \frac{({\rho_E}^T (P^\pi)^t)(s)}{\rho_E(s)} \tag{6.17}$$

Théorème 2 (Borne des performances de CSI)**.** *Soit R^C la récompense retournée par l'algorithme 12. Soient $C_{\pi^*_{\hat{R}^C}}$, ϵ_C et ϵ_R les quantités définies ci-dessus. On a :*

$$0 \leq \mathbb{E}\left[V_{\hat{R}^C}^{\pi^*_{\hat{R}^C}}(s) - V_{\hat{R}^C}^{\pi^E}(s) \middle| s \sim \rho_E \right] \leq \frac{1}{1-\gamma} \left(\epsilon_C \Delta q + \epsilon_R (1 + C_{\pi^*_{\hat{R}^C}}) \right). \tag{6.18}$$

Prenons le temps d'analyser les différences entre cette borne et celle du Théorème 1. Le terme $\bar{\epsilon}_Q$ n'apparaît pas plus dans cette borne que le terme ϵ_R présent ici n'apparaît pour SCIRL. Il faut également se méfier de l'erreur de classification $\epsilon_C^{\rho_E}$. Pour SCIRL, la classification est informée de la structure temporelle du PDM par l'utilisation de l'attribut moyen. Dans CSI, l'introduction de la structure temporelle du PDM n'intervient qu'après l'étape de classification, le terme d'erreur de classification $\epsilon_C^{\rho_E}$ présent ici traite d'une étape "myope", tandis que celui de SCIRL traite d'une approximation de la fonction de qualité de l'expert, paramétrée linéairement grâce à l'attribut moyen.

Le terme Δ_q peut cacher un facteur en $\frac{1}{1-\gamma}$, on retrouve le même type de constantes que dans le Théorème 1.

Enfin, les coefficients de concentration diffèrent légèrement. Celui présent ici est plus petit, il ne correspond pas à un pire des cas comme celui de SCIRL. On peut le voir comme une mesure de la similitude entre π^E et $\pi^*_{\hat{R}^C}$. Si $\pi^*_{\hat{R}^C} \approx \pi^E$ alors $C_{\pi^*_{\hat{R}^C}} \approx 1$ [3].

La preuve, due au travail de Bilal Piot [4], est reportée en annexe A.

Les deux principaux termes d'erreurs présents dans cette borne, ϵ_C et ϵ_R sont conformes aux termes d'erreur usuels en apprentissage supervisé. Utiliser les nombreuses astuces et adaptations rendues possibles par l'usage d'un classifieur et d'un régresseur de l'état de l'art permet d'espérer minimiser ces deux termes. Les résultats théoriques de l'apprentissage supervisé peuvent être facilement intégrés à CSI. Il est plus raisonnable d'espérer contrôler le terme ϵ_R présent dans cette borne que le terme $\bar{\epsilon}_Q$ de la borne du Théorème 1.

Cette borne est également rassurante en ce qui concerne l'apparition de solutions dégénérées ; si les deux termes d'erreur supervisés sont annulés on obtient un résultat similaire à celui du Corollaire 1.

Corollaire 2 (CSI dans le cas idéal)**.** *Si $\rho_E > 0$ et le classifieur et le régresseur sont parfaits ($\epsilon_C = 0$ et $\epsilon_R = 0$), alors π^E est l'*unique *politique optimale pour $\hat{R}^C$.*

Démonstration. La fonction q est la fonction de qualité optimale pour la politique π^C vis-à-vis de la récompense R^C, par définition (voir Équation 6.5). Si $\epsilon_C = 0$, alors $\pi^C = \pi^E$. Cela signifie que $\forall s, \pi^E(s)$ est le seul élément de l'ensemble $\arg\max_{a \in \mathcal{A}} q(s,a)$. Du coup, $\pi^C = \pi^E$ est l'unique politique optimale pour R^C. Et puisque $\epsilon_R = 0$, on a $\hat{R}^C = R^C$, d'où le résultat. □

6.3 *Validation empirique*

La borne théorique donnée à la section précédente ne prend pas l'heuristique proposée Équation 6.10 en compte, puisque celle-ci introduit des échantillons non conformes à l'Équation 6.8. Nous allons donc, comme nous l'avons fait pour SCIRL section 5.4 tester empiriquement la version heuristique de CSI afin de montrer que l'heuristique est valable.

6.3.1 *Introduction*

Nous retrouvons ici les bancs d'essai utilisés section 5.4, le problème du *mountain-car* et celui du *highway*. La classification s'étant avérée moins performante que les méthodes d'ARI, nous nous concentrons ici sur celles-ci, et comparons SCIRL, CSI et RelEnt. Le critère utilisé est le même que celui des expériences précédentes, à savoir un critère donnant une idée de l'optimalité de la politique optimale pour $\hat{R}^C$ par rapport à la récompense inconnue R^E.

L'instanciation de CSI utilisée est celle de l'algorithme 13. Nous préciserons dans les sous-sections le choix du classifieur et du régresseur.

6.3.2 *CSI appliqué au* mountain-car

Le protocole expérimental est le même que dans la sous-section 5.4.2. Le classifieur choisi pour CSI est le même que celui dont les performances sont illustrées Figure 6.1. Il s'agit de d'une SVM [5]. Le noyau est gaussien. Nous sommes dans un cas où l'ingénierie

3. Il est nécessaire de connaître $\hat{R}^C$, le résultat de CSI, pour calculer $C_{\pi^*_{\hat{R}^C}}$. Si l'on souhaite disposer d'une borne à priori alors il est possible de relâcher la borne en utilisant le coefficient $\max_\pi C_\pi$, qui reste plus petit que le coefficient de la borne de SCIRL (Équation 5.13).

4. Co-auteur des contributions sur CSI [Kle+13a ; Kle+13b].

5. Module SVM de Scikit-learn

déployée par l'opérateur est minimale. Le régresseur est choisi dans le même esprit, il s'agit d'une SVR (*Support Vector Regression*, Régresseur à Vecteur Support) [6] utilisée avec ses paramètres par défaut.

6. Module SVR de Scikit-learn

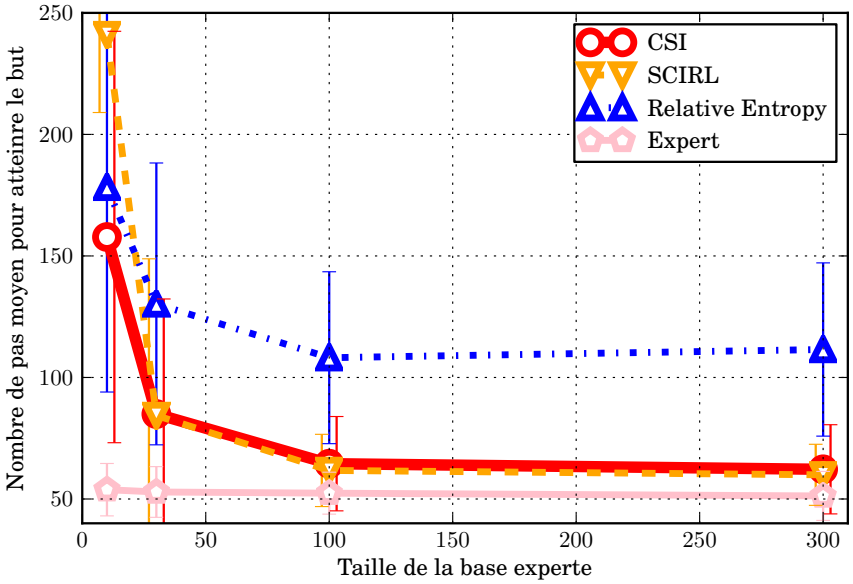


FIGURE 6.1: Algorithmes d'ARI sur le *mountain-car*.

Les résultats obtenus après avoir répété 100 fois l'expérience sont donnés Figure 6.1, il est clair que les deux nouveaux algorithmes d'ARI SCIRL et CSI sont plus performants que RelEnt (qui était déjà lui-même plus performant que l'approche supervisée). Il n'est pas clair en revanche qui de SCIRL ou CSI est le plus performant. Pour en avoir le cœur net, nous allons procéder à un test statistique d'égalité des moyennes. Si un test d'égalité des variances de Bartlett réussit, un test de Student est réalisé, si le Bartlett échoue, c'est un test de Welch. Les p-valeurs sont données table 6.1. On voit que la seule abscisse pour laquelle on peut affirmer que les moyennes sont inégales est la première, où CSI est plus performant que SCIRL (et aussi que RelEnt). Pour les abscisses suivantes, SCIRL et CSI ont peu ou prou les mêmes performances, très proches des performances expertes.

Taille de la base experte	p-valeur
10	$\mathbf{1.5e-12}$
30	$3.8e-01$
100	$1.3e-02$
300	$7.4e-01$

TABLE 6.1: Test d'égalité des moyennes de Student ou de Welch concernant CSI et SCIRL sur le problème du *mountain-car*. Des p-valeurs élevées ($> 1.0 \times 10^{-02}$) signifient que l'hypothèse d'égalité des moyennes ne peut être rejetée.

6.3.3 *CSI appliqué au* highway

Là encore, le protocole expérimental est le même que celui que nous avons suivi sous-section 5.4.3. L'approche supervisée, l'algorithme 1 armé des attributs standard définis Équation 2.9 étant moins performante que les algorithmes d'ARI, nous ne donnons pas ses performances ici. Elle constitue la première étape de CSI. La seconde étape, la régression, est effectuée par un régresseur aux moindres carrés (algorithme 14), utilisant

également les attributs de l'Équation 2.9.

Algorithme 14: Régresseur aux moindres carrés pour CSI

Entrées : Une base d'entraînement D^q_{sar};
Données : Une fonction d'attribut ϕ sur l'espace d'état action;
Sorties : Une approximation $\hat{R}^C$

1 Initialiser la matrice X de taille $N \times d_\phi$:

$$X = \Big(\phi_j(s_i, a_i)\Big)_{i,j}; \tag{6.19}$$

Initialiser le vecteur y de taille $N \times 1$:

$$y_i = \hat{r}_i; \tag{6.20}$$

Calculer le vecteur de paramètres β :

$$\beta = (X^T X)^{-1} X^T y \tag{6.21}$$

retourner *La fonction* $\beta^T \phi$

Les résultats obtenus sont donnés Figure 6.2. Les performances des algorithmes d'ARI étant très proches les unes des autres, nous offrons un grossissement Figure 6.3 qui montre que CSI semble disposer d'un avantage sur les deux autres approches. Cela est confirmé par un test statistique d'égalité des moyennes (comme précédemment, il s'agit d'un test de Student ou de Welch selon qu'un test de Bartlett d'égalité des variances réussit ou non) dont les résultats sont fournis table 6.2 : pour les abscisses de la Figure 6.3, les performances de CSI sont bel et bien supérieures à celles de SCIRL.

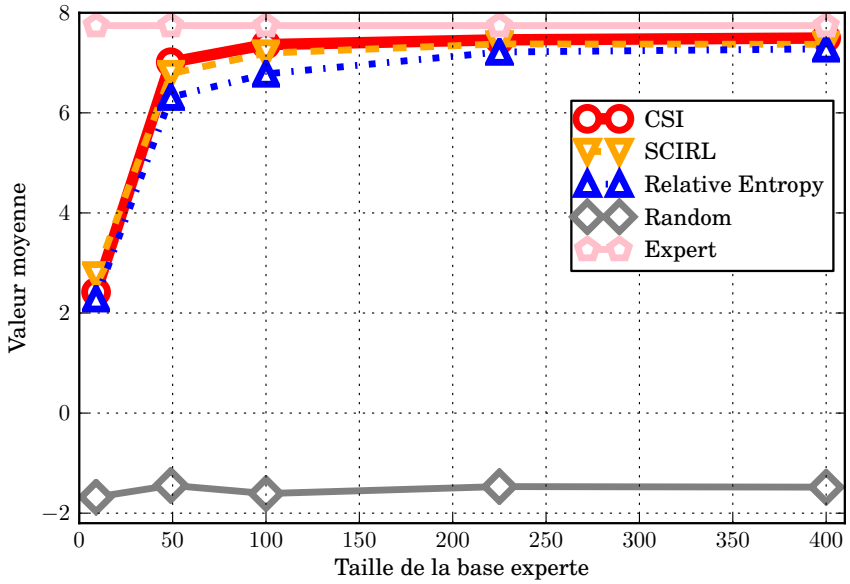


FIGURE 6.2: Algorithmes d'ARI sur le *highway*.

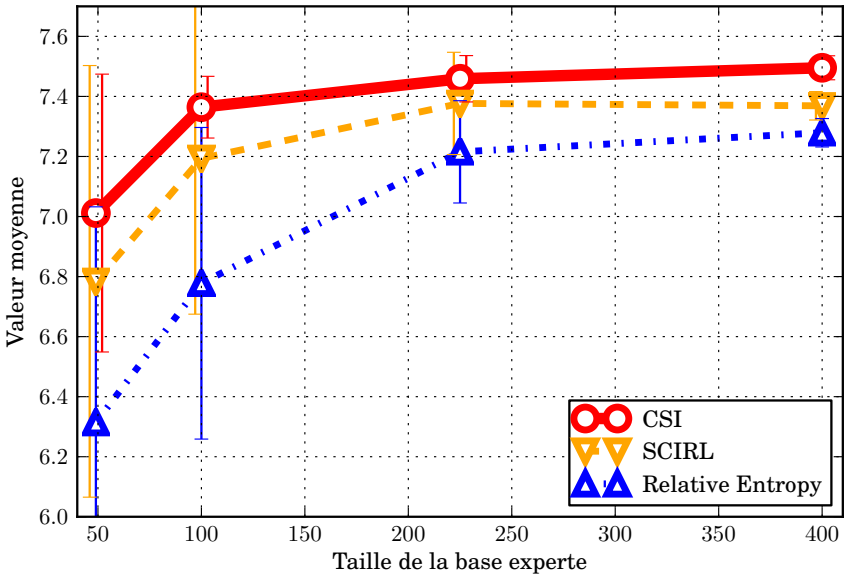


FIGURE 6.3: Grossissement de la Figure 6.2.

Taille de la base experte	p-valeur
9	$3.0e-01$
49	**8.9e − 03**
100	**1.8e − 03**
225	**2.4e − 05**
400	**2.0e − 50**

TABLE 6.2: Test d'égalité des moyennes de Student ou de Welch concernant CSI et SCIRL sur le problème du *highway*. Des p-valeurs élevées ($> 1.0 \times 10^{-02}$) signifient que l'hypothèse d'égalité des moyennes ne peut être rejetées.

6.4 Conclusion

L'algorithme CSI que nous venons de présenter est basé sur un principe commun avec SCIRL, qui consiste à voir la fonction de score d'un classifieur comme une fonction de qualité optimale. L'introduction de la structure temporelle du PDM dans l'approche diffère cependant. L'utilisation cascadée de deux approches supervisée offre en effet des garanties théoriques plus serrées, avec des termes d'erreur plus faciles à maîtriser.

La souplesse offerte par la liberté de choix des deux approches supervisées laisse à l'opérateur la possibilité d'adapter l'approche au problème qu'il cherche à résoudre. Une heuristique est offerte qui permet à CSI de fonctionner avec une expertise humaine minimale en n'utilisant que des données expertes. Cela est illustré sur les mêmes bancs d'essai que ceux qui ont permis la validation empirique de SCIRL. Les performances empiriques offertes par CSI sont légèrement supérieures à celles de SCIRL. Les deux approches sont plus performantes que l'approche supervisée au prix cependant de devoir résoudre le PDM une fois la récompense obtenue.

7
Conclusions et perspectives

7.1 Rappel des contributions

L'un des freins à l'adoption généralisée de la robotique dans la société est le coût élevé des dispositifs automatiques. Du fait de l'augmentation exponentielle des capacités techniques du matériel, le goulot d'étranglement se situe de plus en plus au niveau du coût de conception des systèmes intelligents. Le développement d'algorithmes génériques pouvant s'adapter à une vaste gamme de problèmes est l'un des moyens par lesquels la réduction de ces coûts de développement pourrait être réalisée.

L'AR est un cadre de travail offrant de tels algorithmes. Malheureusement, ils nécessitent pour fonctionner une description formelle de la tâche qu'ils doivent accomplir sous la forme d'une fonction de récompense. La conception d'une bonne fonction de récompense est un problème non trivial nécessitant des réglages fins de la part de l'opérateur.

Savoir effectuer la tâche (ou savoir comment faire, sans pour autant savoir le faire [1]) n'est malheureusement pas suffisant pour pouvoir définir rapidement une bonne fonction de récompense, les capacités d'introspection de l'humain étant notoirement mauvaises.

1. Pour jongler, lancez les balles en l'air sans les laisser tomber au sol.

La nécessité d'un travail non trivial de la part d'un opérateur qualifié rend l'application des algorithmes d'AR difficile voire impossible en dehors d'un laboratoire [2]. Pouvoir établir la fonction de récompense non plus formellement mais à l'aide de démonstrations permettrait, nous l'espérons, la commercialisation de systèmes intuitifs où le client n'aurait qu'à réaliser quelques fois la tâche pour que le robot l'imite.

2. De recherche ou de conception.

Le chapitre 2 a introduit le formalisme de l'apprentissage par imitation et traité dans un premier temps des approches supervisées, qui n'apprennent pas de fonction de récompense mais imitent directement le comportement de l'expert. L'importance du choix des attributs a également été abordée et un état de l'art de l'apprentissage par imitation par le biais d'approches supervisées a été fourni. Dans un second temps, le formalisme des PDM et de l'AR a été introduit, nous permettant de préciser notamment les notions de fonction de récompense et de fonction de valeur. Une fois cela posé, nous avons pu définir le problème de l'ARI, ainsi que les deux critères d'évaluation que nous avons utilisés respectivement dans notre analyse théorique et dans nos évaluations empiriques.

Le chapitre 3 propose un état de l'art des méthodes d'ARI. Nous y avons fait apparaître la nécessité pour la plupart des algorithmes d'ARI de disposer au moins d'un simulateur du système à contrôler. Cette nécessité vient de la structure commune à ces algorithmes, qui implique la résolution répétée du PDM concerné. La quantité d'information nécessaire est bien plus importante que pour les méthodes supervisées étudiées

au chapitre précédent, qui ne nécessitent que des données produites par l'expert. Certaines méthodes récentes d'ARI s'affranchissent de la structure itérative des premières approches, mais la plupart ne peuvent passer à l'échelle, ou nécessitent de connaître les probabilités de transition du PDM. Un algorithme (RelEnt) sort cependant du lot, puisqu'il peut fonctionner en utilisant uniquement des données produites par l'expert et des données produites par une politique aléatoire, afin d'avoir un échantillonnage représentatif du PDM.

Grâce à la première de nos contributions, LSTD-μ (chapitre 4), il est possible de faire fonctionner les algorithmes itératifs de la littérature à partir des mêmes entrées que RelEnt. En effet, LSTD-μ permet l'estimation de l'attribut moyen d'une politique arbitraire en mode *batch* et *off-policy*, en adaptant l'algorithme d'estimation de la valeur LSTD à l'estimation de l'attribut moyen, dont nous remarquons qu'il s'agit de la généralisation vectorielle d'une fonction de valeur. Pour peu que l'on puisse résoudre le PDM avec un algorithme d'AR présentant les mêmes caractéristiques (par exemple LSPI), alors il est possible de trouver la politique optimale pour une récompense arbitraire et d'estimer l'attribut moyen de cette politique grâce à une seule base de données, échantillonnées une fois pour toutes.

Nous proposons chapitre 5 un nouvel algorithme d'ARI, SCIRL, qui diffère du schéma itératif. En utilisant l'attribut moyen de l'expert (qu'il est possible d'estimer en utilisant LSTD-μ, notre contribution du chapitre précédent) pour paramétrer la fonction de score d'un classifieur à marge structurée (décrit au chapitre 2), nous obtenons directement une fonction de récompense. L'analyse théorique que nous fournissons démontre que la politique de l'expert tend à être optimale pour cette fonction de récompense quand l'erreur de classification et l'erreur d'estimation de l'attribut moyen sont petites. Nous proposons une heuristique permettant de s'affranchir de la nécessité de disposer d'une base de données représentative du PDM, et illustrons empiriquement les bonnes performances de SCIRL comparativement à une méthode supervisée et à RelEnt. Les capacités de généralisation des algorithmes d'ARI sont empiriquement supérieures à celles de la méthode supervisée. Il faut cependant tenir compte du fait que la sortie d'un algorithme d'ARI est une récompense, qu'il faut optimiser pour obtenir une politique de contrôle tandis que les algorithmes supervisés renvoient directement une politique.

Plus souple, l'algorithme CSI présenté au chapitre 6 ne nécessite pas d'estimer l'attribut moyen de l'expert, et consiste en la cascade de deux méthode supervisées : une étape de classification permettant d'apprendre une fonction de score et une étape de régression permettant, grâce à une inversion de l'équation de Bellman, d'apprendre la récompense associée à cette fonction de score. Cet algorithme dispose de garanties théoriques : la politique de l'expert est optimale pour la récompense apprise si les erreurs de classification et de régression sont petites. La panoplie d'approches de regression et de classification fournie par la littérature nous permet de contrôler ces deux termes plus facilement que ceux présents dans la borne de SCIRL. Nous proposons une nouvelle heuristique, qui là encore nous permet de nous affranchir de la nécessité de données autres que celles échantillonnées par l'expert, comme l'illustrent les expériences que nous menons. Sur ces expériences, les performances de CSI sont légèrement supérieures à celles de SCIRL.

Nous démontrons empiriquement que les deux algorithmes d'ARI que nous proposons peuvent fonctionner avec peu de données. Bien que les problèmes sur lesquels nous les testons soient des problèmes jouet, nos algorithmes sont à notre connaissance les seuls algorithmes d'ARI pouvant fonctionner uniquement avec des données

expertes[3]. Les résultats obtenus montrent des performances supérieures à celles des algorithmes supervisés et à une approche récente d'ARI ; il faut cependant trouver le moyen d'optimiser la récompense pour obtenir une politique tandis que les algorithmes supervisés la fournissent directement.

3. Moyennant l'utilisation d'heuristiques

Les systèmes réels sont souvent fragiles et interagir avec peut être coûteux. Il est également difficile de les simuler. On comprend alors l'intérêt de développer des algorithmes moins gourmands en données.

7.2 *Perspectives de recherche*

Les deux nouveaux algorithmes d'ARI que nous proposons disposent de garanties théoriques et une étude empirique illustre le fait qu'ils sont plus performants que les méthodes de l'état de l'art. Il serait souhaitable de renforcer cette étude empirique en les appliquant à d'autres problèmes, que ce soit des problèmes déjà résolus par d'autres méthodes, à des fins comparatives, ou des problèmes ouverts. Notamment, deux domaines ont commencé à être explorés, mais les travaux n'ont pu être menés à terme et de fait n'apparaissent pas dans ce manuscrit. Le premier est l'*Arcade Learning Environment* (ALE) : il s'agit d'un émulateur d'Atari 2600 permettant de jouer à une vaste collection de jeux vidéos. Le second est l'application d'algorithmes d'ARI à la commande d'un bras robot à partir de signaux issus d'une interface cerveau/machine (*Brain Computer Interface*, BCI)[4]. Dans ces deux cas, la difficulté n'est pas dans l'application des algorithmes d'ARI eux-même, mais dans l'optimisation de la récompense qu'ils trouvent afin d'obtenir une politique de contrôle. Nous avons dans nos travaux considéré que cette étape ne posait pas problème, afin de nous concentrer sur le problème de l'ARI tel qu'envisagé par la communauté. L'application à des problèmes pratiques doit faire face à l'optimisation de la récompense, il faut alors faire appel à des méthodes de l'état de l'art en AR et des adaptations *ad-hoc* au problème (par exemple dans le choix des attributs).

4. Voir la section C.4 pour une description des travaux effectués sur la BCI.

Pour remédier de manière plus générale à ce problème[5], nous pourrions également tenter de développer un cadre pour les algorithmes d'ARI dans lequel l'optimisation finale de la récompense est simplifiée ou contournée. La politique de classification du classifieur paramétré par l'attribut moyen de l'expert qui constitue SCIRL est par exemple une bonne politique de départ pour un algorithme d'AR en ligne. Les données nécessaires à l'apprentissage de la politique de contrôle (dont SCIRL n'a pas besoin, grâce à son heuristique, pour trouver la récompense) pourraient alors être recueillies par la politique de classification, qui convergerait alors petit à petit grâce à l'algorithme d'AR en ligne vers la politique optimale pour la récompense trouvée par SCIRL. La politique issue de la première étape supervisée de CSI pourrait être utilisée de la même manière. Il est également possible de poursuivre l'approche envisagée dans [Gei+13a], consistant à instancier CSI avec une SVM et une SVR et à réunir les deux problèmes d'optimisation en un seul, dont la résolution permettrait d'obtenir directement une politique prenant en compte la structure de la récompense.

5. Qui n'affecte pas les approches supervisées.

Finalement, il faudrait intégrer l'AR et l'ARI dans des systèmes informatiques réels aux mains d'utilisateurs non qualifiés. Cela nécessitera sans doute d'importants travaux concernant l'intégration de multiples algorithmes dans un schéma hiérarchique : avant d'apprendre à un robot à faire un mojito, il faut qu'il apprenne à saisir un verre et une bouteille. Mais lorsqu'on lui apprend la recette, on ne veut pas avoir à se soucier des détails de la saisie manuelle d'objet, on fonctionne à un niveau d'abstraction plus élevé.

Il faut donc qu'une politique par exemple "Saisir un verre" faisant appel à des actions telles que "Fermer doigt 1 de la main gauche" et "Tourner poignet gauche dans le sens direct" puisse être utilisée directement comme une action par une politique de plus haut niveau, à côté d'actions telles que "Piler la glace" et "Placer la rondelle de citron". Il faudra également effectuer des progrès dans le domaine de l'interface homme/machine pour parvenir à gérer intuitivement des systèmes aussi complexes. Au-delà de l'aspect robotique grand public, les chercheurs et ingénieurs ont eux aussi beaucoup à gagner à manipuler intuitivement les machines, mais cela n'est pas facilement compatible avec la nécessité de rapidement changer de multiples paramètres.

A
Démonstrations

A.1 Borne sur la performance de SCIRL

Avant de démontrer le Théorème 1, rappelons un résultat dont nous allons avoir besoin :

Lemme 1 ([Muno7, Lemme 4.2]). *Soit A une matrice inversible telle que tous les éléments de A^{-1} sont positifs. Les solutions de l'inégalité $Au \leq b$ sont aussi solution de $u \leq A^{-1}b$.*

Preuve du Théorème 1. Nous allons décomposer le terme $V_{R_\theta}^{\pi^*_{R_\theta}} - V_{R_\theta}^{\pi^E}$ (qui est positif ou, au mieux, nul, puisque $\pi^*_{R_\theta}$ est optimale) afin de faire apparaître le résidu de Bellman $\mathbb{E}\left[B^*_{R_\theta} V_{R_\theta}^{\pi^E}(s) - V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E\right]$. On rappelle que les définitions des opérateurs d'optimalité et d'évaluation de Bellman B^*_R et B^π_R sont données équations 2.46 et 2.39.

$$V_{R_\theta}^{\pi^*_{R_\theta}} - V_{R_\theta}^{\pi^E} = V_{R_\theta}^{\pi^*_{R_\theta}} - B_{R_\theta}^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^E} + B_{R_\theta}^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^E} - B^*_{R_\theta} V_{R_\theta}^{\pi^E} + B^*_{R_\theta} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E} \tag{A.1}$$

or $B_{R_\theta}^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^E} \leq B^*_{R_\theta} V_{R_\theta}^{\pi^E}$, donc $B_{R_\theta}^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^E} - B^*_{R_\theta} V_{R_\theta}^{\pi^E} \leq 0$, donc :

$$V_{R_\theta}^{\pi^*_{R_\theta}} - V_{R_\theta}^{\pi^E} \leq V_{R_\theta}^{\pi^*_{R_\theta}} - B_{R_\theta}^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^E} + B^*_{R_\theta} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E} \tag{A.2}$$

$$= B_{R_\theta}^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^*_{R_\theta}} - B_{R_\theta}^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^E} + B^*_{R_\theta} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E} \tag{A.3}$$

$$= R_\theta + \gamma P^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^*_{R_\theta}} - R_\theta - \gamma P^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^E} + B^*_{R_\theta} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E} \tag{A.4}$$

$$= \gamma P^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^*_{R_\theta}} - \gamma P^{\pi^*_{R_\theta}} V_{R_\theta}^{\pi^E} + B^*_{R_\theta} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E} \tag{A.5}$$

$$= \gamma P^{\pi^*_{R_\theta}} \left(V_{R_\theta}^{\pi^*_{R_\theta}} - V_{R_\theta}^{\pi^E}\right) + B^*_{R_\theta} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E}, \tag{A.6}$$

$$\Rightarrow (I - \gamma P^{\pi^*_{R_\theta}})(V_{R_\theta}^{\pi^*_{R_\theta}} - V_{R_\theta}^{\pi^E}) \leq B^*_{R_\theta} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E} \tag{A.7}$$

En vertu de [Muno7, Lemme 4.2], car $(I - \gamma P^{\pi^*_{R_\theta}}) = \sum_{t\geq 0} (P^{\pi^*_{R_\theta}})^t$ a tous ses éléments positifs,

$$V_{R_\theta}^{\pi^*_{R_\theta}} - V_{R_\theta}^{\pi^E} \leq (I - \gamma P^{\pi^*_{R_\theta}})^{-1} (B^*_{R_\theta} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E}). \tag{A.8}$$

Comme $(I - \gamma P^{\pi^*_{R_\theta}})^{-1} = \sum\limits_{t\geq 0} \gamma^t (P^{\pi^*_{R_\theta}})^t$,

$$V_{R_\theta}^{\pi^*_{R_\theta}} - V_{R_\theta}^{\pi^E} \leq \sum_{t\geq 0} \gamma^t (P^{\pi^*_{R_\theta}})^t (B^*_{R_\theta} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E}). \tag{A.9}$$

D'après la définition de $c(t)$ (Équation 5.14) et du coefficient de concentration C_f (Équation 5.13) :

$$0 \leq \mathbb{E}\left[V_{R_\theta}^{\pi^*_{R_\theta}}(s) - V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E \right] \leq \sum_{t\geq 0} c(t) \mathbb{E}\left[B_{R_\theta}^* V_{R_\theta}^{\pi^E}(s) - V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E \right] \quad \text{(A.10)}$$

$$0 \leq \mathbb{E}\left[V_{R_\theta}^{\pi^*_{R_\theta}}(s) - V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E \right] \leq \frac{C_f}{1-\gamma} \mathbb{E}\left[B_{R_\theta}^* V_{R_\theta}^{\pi^E}(s) - V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E \right] \quad \text{(A.11)}$$

$$\text{(A.12)}$$

Il nous faut maintenant borner le résidu de Bellman

$$\mathbb{E}\left[B_{R_\theta}^* V_{R_\theta}^{\pi^E}(s) - V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E \right], \quad \text{(A.13)}$$

que nous décomposons en :

$$B_{R_\theta}^* V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E} = B_{R_\theta}^* V_{R_\theta}^{\pi^E} - B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E} + B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E}. \quad \text{(A.14)}$$

Nous allons borner séparément

$$\mathbb{E}\left[B_{R_\theta}^* V_{R_\theta}^{\pi^E}(s) - B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E \right] \text{ et } \mathbb{E}\left[B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E}(s) - V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E \right]. \quad \text{(A.15)}$$

La politique de classification π^C est gloutonne vis-à-vis de $q_\theta = \theta^T \hat{\mu}^{\pi^E}$. Donc, pour chaque couple état-action $(s,a) \in \mathcal{S} \times \mathcal{A}$:

$$q_\theta(s, \pi^C(s)) \geq q_\theta(s,a) \Leftrightarrow Q_{R_\theta}^{\pi^E}(s,a) \leq Q_{R_\theta}^{\pi^E}(s, \pi^C(s)) + \epsilon_Q(s, \pi^C(s)) - \epsilon_Q(s,a). \quad \text{(A.16)}$$

Par définition, $Q_{R_\theta}^{\pi^E}(s,a) = [B_{R_\theta}^a V_{R_\theta}^{\pi^E}](s)$ et $Q_{R_\theta}^{\pi^E}(s, \pi^C(s)) = [B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E}](s)$. Donc, pour $s \in \mathcal{S}$:

$$\forall a \in A,\ [B_{R_\theta}^a V_{R_\theta}^{\pi^E}](s) \leq [B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E}](s) + \epsilon_Q(s, \pi^C(s)) - \epsilon_Q(s,a) \quad \text{(A.17)}$$

$$\Rightarrow [B_{R_\theta}^* V_{R_\theta}^{\pi^E}](s) \leq [B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E}](s) + \max_{a\in\mathcal{A}} \epsilon_Q(s,a) - \min_{a\in\mathcal{A}} \epsilon_Q(s,a). \quad \text{(A.18)}$$

En passant à l'espérance selon ρ_E et tout en rappelant que $B_{R_\theta}^* V_{R_\theta}^{\pi^E} \geq V_{R_\theta}^{\pi^E}$, le premier terme est borné :

$$0 \leq \mathbb{E}\left[B_{R_\theta}^* V_{R_\theta}^{\pi^E}(s) - B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E \right] \leq \bar{\epsilon}_Q. \quad \text{(A.19)}$$

Il reste finalement à borner le terme $\mathbb{E}\left[B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E}(s) - V_{R_\theta}^{\pi^E}(s) \middle| s \sim \rho_E \right]$. À cet effet nous introduisons $M \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ la matrice diagonale définie par $M = \text{diag}(\mathbb{1}(\pi^C(s) \neq \pi^E(s)))$. Grâce à cela, l'opérateur de Bellman $B_R^{\pi^C}$ peut s'écrire, pour tout $V \in \mathbb{R}^{\mathcal{S}}$:

$$B_R^{\pi^C} V = R + \gamma M P^{\pi^C} V + \gamma (I - M) P^{\pi^E} V = R + \gamma P^{\pi^E} V + \gamma M (P^{\pi^C} - P^{\pi^E}) V. \quad \text{(A.20)}$$

En appliquant cet opérateur à $V_{R_\theta}^{\pi^E}$ et en utilisant le fait que $R + \gamma P^{\pi^E} V_{R_\theta}^{\pi^E} = B_{R_\theta}^{\pi^E} V_{R_\theta}^{\pi^E} = V_{R_\theta}^{\pi^E}$, nous obtenons :

$$B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E} = \gamma M (P^{\pi^C} - P^{\pi^E}) V_{R_\theta}^{\pi^E} \quad \text{(A.21)}$$

$$\Rightarrow |\rho_E{}^\top (B_{R_\theta}^{\pi^C} V_{R_\theta}^{\pi^E} - V_{R_\theta}^{\pi^E})| = \gamma |\rho_E{}^T M (P^{\pi^C} - P^{\pi^E}) V_{R_\theta}^{\pi^E}|. \quad \text{(A.22)}$$

D'après l'inégalité de Hoelder [1] appliquée ici avec $p = 1$ et $q = \infty$:

$$|\rho_E{}^T M (P^{\pi^C} - P^{\pi^E}) V_{R_\theta}^{\pi^E}| \leq \|\rho_E{}^T M\|_1 \|(P^{\pi^C} - P^{\pi^E}) V_{R_\theta}^{\pi^E}\|_\infty \quad \text{(A.23)}$$

$$= \epsilon_C^{\rho_E} \|(P^{\pi^C} - P^{\pi^E}) V_{R_\theta}^{\pi^E}\|_\infty \quad \text{(A.24)}$$

1. Qui stipule que si $1 \leq p, q \leq \infty$, $\frac{1}{p} + \frac{1}{q} = 1$, alors $|< f, g >| \leq \|f\|_p \|g\|_q$.

Bornons le terme $(P^{\pi^C} - P^{\pi^E})V_{R_\theta}^{\pi^E}$ pour un état quelconque :

$$(P^{\pi^C} - P^{\pi^E})V_{R_\theta}^{\pi^E}(s) = |\sum_{s'}\left(p\left(s'|s,\pi^C(s)\right) - p\left(s'|s,\pi^E(s)\right)\right)V_{R_\theta}^{\pi^E}(s)| \tag{A.25}$$

$$\leq \sum_{s'}|\left(p\left(s'|s,\pi^C(s)\right) - p\left(s'|s,\pi^E(s)\right)\right)||V_{R_\theta}^{\pi^E}(s)| \tag{A.26}$$

$$= \sum_{s'}|\left(p\left(s'|s,\pi^C(s)\right) - p\left(s'|s,\pi^E(s)\right)\right)|\frac{\|R_\theta\|_\infty}{1-\gamma} \tag{A.27}$$

$$\leq \left(\sum_{s'}p\left(s'|s,\pi^C(s)\right) + \sum_{s'}p\left(s'|s,\pi^E(s)\right)\right)\frac{\|R_\theta\|_\infty}{1-\gamma} \tag{A.28}$$

$$\leq \frac{2}{1-\gamma}\|R_\theta\|_\infty. \tag{A.29}$$

Ce terme ne dépend pas de s donc la norme infinie est identique, et finalement on a $\|(P^{\pi^C} - P^{\pi^E})V_{R_\theta}^{\pi^E}\|_\infty \leq \frac{2}{1-\gamma}\|R_\theta\|_\infty$, ce qui permet de borner le dernier terme

$$\left|\mathbb{E}\left[B_{R_\theta}^{\pi^C}V_{R_\theta}^{\pi^E}(s) - V_{R_\theta}^{\pi^E}(s)\middle| s\sim\rho_E\right]\right| \leq \epsilon_C^{\rho_E}\frac{2\gamma}{1-\gamma}\|R_\theta\|_\infty. \tag{A.30}$$

Injecter les bornes des équations (A.19) et (A.30) dans l'équation (A.12) achève la démonstration. □

A.2 Borne sur la performance de CSI

Preuve du Théorème 2. Nous allons borner l'expression :

$$\mathbb{E}\left[V_{\hat{R}^C}^{\pi^*_{\hat{R}^C}}(s) - V_{\hat{R}^C}^{\pi^E}(s)\middle| s\sim\rho_E\right] \tag{A.31}$$

qui est positive ou, au mieux, nulle puisque $\pi^*_{\hat{R}^C}$ est optimale. Pour cela, décomposons le terme dans l'espérance :

$$V_{\hat{R}^C}^{\pi^*_{\hat{R}^C}} - V_{\hat{R}^C}^{\pi^E} = (V_{\hat{R}^C}^{\pi^*_{\hat{R}^C}} - V_{R^C}^{\pi^*_{\hat{R}^C}}) + (V_{R^C}^{\pi^*_{\hat{R}^C}} - V_{R^C}^{\pi^E}) + (V_{R^C}^{\pi^E} - V_{\hat{R}^C}^{\pi^E}). \tag{A.32}$$

Nous allons borner ces trois termes en deux fois.

Pour toute politique π,

$$\rho_E{}^T(V_{R^C}^{\pi} - V_{\hat{R}^C}^{\pi}) = \rho_E{}^T V_{(R^C-\hat{R}^C)}^{\pi} \tag{A.33}$$

$$= \rho_E{}^T(I-\gamma P^\pi)^{-1}\epsilon_\pi^R \tag{A.34}$$

$$\leq \frac{C_\pi}{1-\gamma}\rho_E{}^T\epsilon_\pi^R \tag{A.35}$$

$$\leq \frac{C_\pi}{1-\gamma}\|\epsilon_\pi^R\|_{1,\rho_E}. \tag{A.36}$$

Le passage de la deuxième à la troisième ligne vient de la décomposition en série entière de $(I-\gamma P^\pi)^{-1}$, et le passage à la dernière ligne est possible car l'une des propriétés de la norme L_1 ρ-pondérée est que $\forall X, \rho^T X \leq \|X\|_{1,\rho}$.

Nous sommes maintenant en mesure de borner le premier et dernier terme :

$$\rho_E{}^T(V_{R^C}^{\pi^E} - V_{\hat{R}^C}^{\pi^E}) \leq \frac{C_{\pi^E}}{1-\gamma}\epsilon_R \tag{A.37}$$

$$= \frac{1}{1-\gamma}\epsilon_R \tag{A.38}$$

$$\text{et } \rho_E{}^T(V_{\hat{R}^C}^{\pi^*_{\hat{R}^C}} - V_{R^C}^{\pi^*_{\hat{R}^C}}) \leq \frac{C_{\pi^*_{\hat{R}^C}}}{1-\gamma}\epsilon_R \tag{A.39}$$

$$\Rightarrow \rho_E{}^T\left((V_{R^C}^{\pi^E} - V_{\hat{R}^C}^{\pi^E}) + (V_{\hat{R}^C}^{\pi^*_{\hat{R}^C}} - V_{R^C}^{\pi^*_{\hat{R}^C}})\right) \leq \frac{1 + C_{\pi^*_{\hat{R}^C}}}{1-\gamma}\epsilon_R \tag{A.40}$$

Il nous reste maintenant à trouver une borne pour le deuxième terme. Dans cet objectif, nous introduisons la récompense :

$$\bar{R^C}(s,a) = q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p\left(s'|s,a\right) q(s', \pi^E(s')), \tag{A.41}$$

ce qui nous permet de décomposer le deuxième terme en :

$$V_{R^C}^{\pi^*_{\hat{R}^C}} - V_{R^C}^{\pi^E} = (V_{R^C}^{\pi^*_{\hat{R}^C}} - V_{R^C}^{\pi^C}) + (V_{R^C}^{\pi^C} - V_{\bar{R^C}}^{\pi^E}) + (V_{\bar{R^C}}^{\pi^E} - V_{R^C}^{\pi^E}). \tag{A.42}$$

En conséquence directe de leurs définitions, π^C est optimale pour R^C, donc :

$$V_{R^C}^{\pi^*_{\hat{R}^C}} - V_{R^C}^{\pi^C} \leq 0, \tag{A.43}$$

et conséquemment :

$$V_{R^C}^{\pi^*_{\hat{R}^C}} - V_{R^C}^{\pi^E} \leq (V_{R^C}^{\pi^C} - V_{\bar{R^C}}^{\pi^E}) + (V_{\bar{R^C}}^{\pi^E} - V_{R^C}^{\pi^E}). \tag{A.44}$$

Toujours par définition, nous avons $V_{R^C}^{\pi^C} = q(\cdot, \pi^C(\cdot))$ et $V_{\bar{R^C}}^{\pi^E} = q(\cdot, \pi^E(\cdot))$, donc :

$$\rho_E{}^T(V_{R^C}^{\pi^C} - V_{\bar{R^C}}^{\pi^E}) = \rho_E{}^T(q(\cdot, \pi^C(\cdot)) - q(\cdot, \pi^E(\cdot))) \tag{A.45}$$

$$= \sum_{s \in S} \rho_E(s)(q(s, \pi^C(s)) - q(s, \pi^E(s)))\mathbb{1}(\pi^C(s) \neq \pi^E(s)) \tag{A.46}$$

$$\leq \Delta q \sum_{s \in S} \rho_E(s)[\mathbb{1}_{\{\pi^C(s) \neq \pi^E(s)\}}] \tag{A.47}$$

$$\leq \Delta q \epsilon_C. \tag{A.48}$$

Pour conclure, constatons que :

$$\bar{R^C}(s, \pi^E(s)) - R^C(s, \pi^E(s)) = (q(s, \pi^E(s)) - \gamma P^{\pi^E} q(s, \pi^E(s))) \tag{A.49}$$

$$- (q(s, \pi^E(s)) - \gamma P^{\pi^E} q(s, \pi^C(s))) \tag{A.50}$$

$$= \gamma P^{\pi^E} q(s, \pi^C(s)) - \gamma P^{\pi^E} q(s, \pi^E(s))) \tag{A.51}$$

$$= \gamma P^{\pi^E}(q(s, \pi^C(s)) - q(s, \pi^E(s))), \tag{A.52}$$

ce qui nous permet de borner le dernier terme (de l'extension du deuxième terme) :

$$\rho_E{}^T(V_{\bar{R^C}}^{\pi^E} - V_{R^C}^{\pi^E}) = \rho_E{}^T(I - \gamma P^{\pi^E})^{-1}(\bar{R^C}(\cdot, \pi^E(\cdot)) - R^C(\cdot, \pi^E(\cdot))) \tag{A.53}$$

$$= \rho_E{}^T(I - \gamma P^{\pi^E})^{-1}\gamma P^{\pi^E}(q(\cdot, \pi^C(\cdot)) - q(\cdot, \pi^E(\cdot))) \tag{A.54}$$

$$= \frac{\gamma}{1-\gamma}\rho_E{}^T(q(\cdot, \pi^C(\cdot)) - q(\cdot, \pi^E(\cdot))) \tag{A.55}$$

$$\leq \frac{\gamma}{1-\gamma}\Delta q \epsilon_C. \tag{A.56}$$

Finalement la borne pour le deuxième terme est :

$$\rho_E{}^T(V_{R^C}^{\pi^*_{\hat{R}^C}} - V_{R^C}^{\pi^E_C}) \leq (\Delta q + \frac{\gamma}{1-\gamma}\Delta q)\epsilon_C \tag{A.57}$$

$$\leq \frac{\Delta q}{1-\gamma}\epsilon_C. \tag{A.58}$$

En combinant les équations A.40 et A.58, nous obtenons la borne du théorème. □

B
Problèmes jouet

Pour accompagner les explications nous étudierons trois problèmes jouet. Ces problèmes sont suffisamment simples pour qu'une interprétation des grandeurs en jeu soit possible, mais suffisamment complexes pour présenter un défi aux approches existantes. Ils serviront également de bancs d'essai lors de nos études empiriques des nouveaux algorithmes que nous proposons.

B.1 Problème du pendule inversé

Le problème du pendule inversé consiste à maintenir en équilibre une masse située au bout d'un bras en commandant les mouvements du chariot à la base du bras. Il s'agit d'un problème continu, la position du pendule étant décrite par l'angle du bras, à valeurs dans $[-\frac{\pi}{2}, \frac{\pi}{2}]$ et sa vitesse à valeurs dans $\mathbb{R}$.

Ce problème a été utilisé comme banc de test de l'algorithme d'AR LSPI. Lagoudakis et Parr [LP03] en décrivent la dynamique, basée sur la résolution des équations différentielles du mouvement par la méthode d'intégration d'Euler.

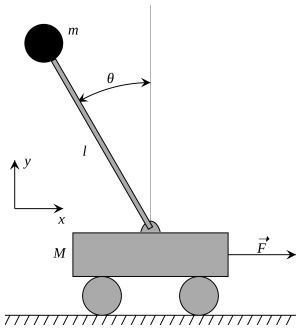


Figure B.1: Schéma de principe du pendule inversé : le chariot est libre de bouger selon l'axe x, le but est de maintenir le pendule en équilibre en commandant le chariot.

– On trouve l'accélération du pendule en utilisant les équations du mouvement de Newton.

$$a = \frac{g \cdot \sin(\theta) - \frac{m \cdot l \cdot \dot{\theta}^2 \cdot \sin(2 \cdot \theta)}{2(m+M)} - \frac{\cos(\theta) \cdot F}{(m+M)}}{4 \cdot \frac{l}{3} - \frac{m \cdot l \cdot \cos(\theta)^2}{m+M}} \tag{B.1}$$

– A chaque pas de temps, on trouve la nouvelle position avec :

$$\theta \leftarrow \theta + \dot{\theta} \cdot \Delta t, \Delta t = 0.1s \tag{B.2}$$

– La vitesse est ensuite également mise à jour :

$$\dot{\theta} \leftarrow \dot{\theta} + a \cdot \Delta t, \Delta t = 0.1s \tag{B.3}$$

L'expert pour ce problème est un agent entraîné avec LSPI (voir aussi sous-section 2.2.3) ; il parvient à maintenir le pendule en équilibre indéfiniment, mais la simulation s'arrête après 3000 pas de temps, soit 5 minutes de temps simulé.

B.2 Problème du mountain-car

Dans le problème du *mountain-car*, une voiture placée dans un creux doit gravir une côte trop pentue pour qu'elle puisse la surmonter du premier coup. La solution du problème consiste à reculer, donc à s'éloigner de l'objectif, en remontant la pente opposée puis à profiter de l'élan pour gravir la pente. Le processus peut-être répété si besoin. Cette nécessité contre-intuitive de devoir s'éloigner de l'objectif pour pouvoir l'atteindre fait de ce problème jouet un bon test pour les algorithmes d'AR. Il a été proposé pour

la première fois par MOORE [Moo90] et a été popularisé lors son introduction dans le livre de SUTTON et BARTO [SB98].

La Figure 1.2 décrit les variables qui représentent l'information dont dispose l'agent pour ce problème. Il y a bien sûr la position de la voiture : le point le plus à gauche a une coordonnée de -1.2, l'objectif, situé le plus à droite, se trouve à la coordonnée 0.6. Le fond de la vallée à la coordonnée -0.5. La deuxième information est la vitesse de la voiture, elle peut aller de -0.07 à 0.07 où un chiffre positif désigne naturellement un déplacement vers la droite et négatif vers la gauche. La vitesse est plafonnée, et si la voiture atteint l'extrémité gauche, sa vitesse est annulée et elle reste en place.

La trajectoire de l'expert représentée Figure 1.2 débute dans le quadrant Sud-Ouest de l'espace d'état, correspondant à une position éloignée de l'objectif et une vitesse dans la direction opposée à l'objectif. La dynamique de l'expert renverse ensuite la vitesse (quadrant Nord-Ouest) rapprochant la voiture de l'objectif (quadrant Nord-Est), jusqu'à ce qu'elle l'atteigne. Dans nos études empiriques, les démonstrations de l'expert suivront ce modèle : le point de départ sera choisi aléatoirement dans le quadrant Sud-Ouest [1], et donc la dynamique entraînera l'exploration de ce quadrant et des deux quadrants Nord. Le quadrant Sud-Est, lui, sera ignoré par l'expert. Les algorithmes d'imitation ne disposeront donc pas d'information concernant le comportement de l'expert dans cet partie de l'espace d'état. La performance de la politique de l'agent imitant l'expert sera bien évidemment testée sur tout l'espace d'état, afin de voir comment l'agent a été capable d'extrapoler.

1. Plus précisément dans une partie du quadrant Sud-Ouest telle que l'expert n'a pas à passer dans le quadrant Sud-Est pour atteindre l'objectif.

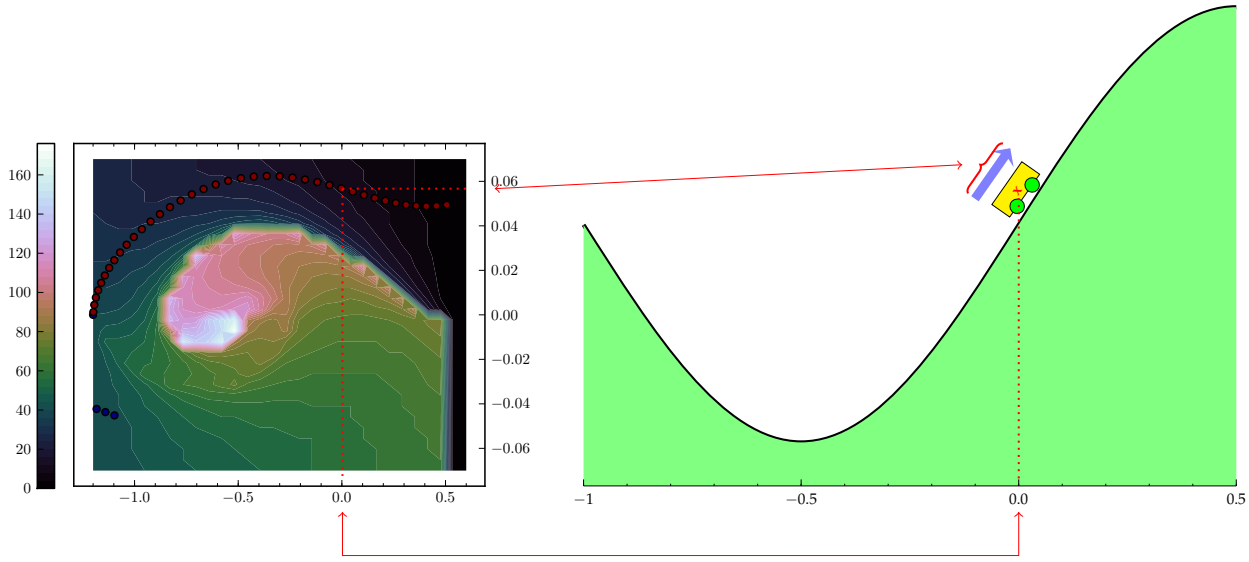


FIGURE B.2: L'espace d'état du *mountain-car* : la position de la voiture est représentée en abscisse et sa vitesse en ordonnée. Représenté sur la figure de gauche par des points : un exemple d'une trajectoire d'une bonne politique de contrôle où l'expert poursuit la montée de la pente opposée à l'objectif avant de la redescendre et de parvenir à gravir la pente à droite. La couleur des points indique l'action effectuée par l'expert : en bleu, il accélère vers la gauche, en rouge vers la droite. La surface en couleur représente le nombre de pas de temps nécessaire à cet expert pour parvenir à l'objectif. La forme en escargot est caractéristique de ce problème, nous la retrouvons Figure 2.2.

B.3 *Problème du* highway

Utilisé sous diverses variantes comme banc d'essai de plusieurs algorithmes d'ARI [AN04 ; SS08 ; SBS08], le simulateur de conduite sur autoroute place l'agent au volant d'une voiture (bleue) plus rapide que le trafic, et qui de fait doit éviter les voitures plus lentes (rouges) présentes sur la route. La voiture peut choisir parmi trois voies de circulation, et peut également rouler dans les deux bas-côtés de chaque côté de la route. Il y a trois vitesses possibles (toutes plus élevées que celles des voitures à éviter), une seule autre voiture est visible, quand l'agent la dépasse une autre apparaît dans

une voie choisie aléatoirement par le simulateur. À la vitesse maximale, l'agent ne peut éviter la voiture qui arrive si elle est située dans la même voie que lui. À chaque instant, il peut choisir de se décaler à droite ou à gauche, ou d'accélérer ou ralentir, il peut également ne rien changer.

Ce simulateur est intéressant car beaucoup de comportements différents sont envisageables, pouvant présenter un subtil compromis entre vitesse et sécurité par exemple. Il est possible de conduire raisonnablement sans aucune collision et sans sortir de la route, à condition de ne pas aller vite. Il est également possible d'aller vite et d'éviter les collisions, à condition que l'on accepte de sortir de la route. Arriver à identifier les motivations précises d'un expert en observant sa conduite est donc un défi.

Contrairement au problème du *mountain-car*, l'espace d'état est de dimension trop large pour être représenté. On y trouve en effet la vitesse de l'agent, sa position horizontale, et la position verticale et horizontale de l'autre voiture.

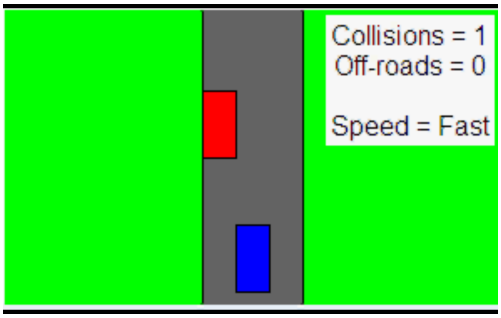


FIGURE B.3: Capture d'écran du problème du *highway*.

C
Contributions

L'évaluation d'une méthode d'apprentissage par imitation peut se faire selon deux critères :

- la nature et la quantité d'information qu'il faut lui donner pour qu'elle fonctionne ;
- la qualité du contrôle qu'elle offre, comparée à la qualité du contrôle de l'expert ayant effectué la démonstration.

Les méthodes d'apprentissage numérique supervisé excellent quant au premier critère, elles ne nécessitent pour fonctionner que de disposer d'une base d'exemples du comportement expert. Les méthodes d'apprentissage par renforcement inverse de l'état de l'art ne sont pas aussi souples. Elles nécessitent [1] pour fonctionner au moins un simulateur du système dynamique à contrôler. Les contributions de cette thèse visent à s'affranchir de ce besoin. Les méthodes d'ARI sont intrinsèquement plus difficiles à utiliser en pratique que les méthodes d'imitation supervisée car après avoir trouvé une récompense, il faut l'optimiser pour obtenir une politique de contrôle. Nous verrons cependant que si l'on peut surmonter cette contrainte, alors les méthodes d'ARI que nous proposons ici présentent expérimentalement une meilleure capacité de généralisation que les méthodes supervisées.

1. Sauf une à notre connaissance, RelEnt, que nous utiliserons de fait comme point de comparaison.

C.1 LSTD-μ

La première contribution, l'algorithme LSTD-μ, auquel le chapitre 4 est consacré, permet d'estimer une grandeur centrale en ARI, l'attribut moyen, sans avoir à utiliser un simulateur du système :

- Elle a donné lieu à une première communication dans un atelier d'IJCAI portant sur l'apprentissage machine grâce à des instructeurs humains :

E. Klein, M. Geist et O. Pietquin. « Batch, Off-policy and Model-Free Apprenticeship Learning ». Dans : *IJCAI Workshop on Agents Learning Interactively from Human Teachers (ALIHT 2011)*. Barcelona (Spain), juil. 2011

- Nous avons ensuite rencontré la communauté d'apprentissage francophone aux JFPDA :

E. Klein, M. Geist et O. Pietquin. « Apprentissage par imitation étendu au cas batch, off-policy et sans modèle ». Dans : *Sixièmes Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2011)*. Rouen (France), juin 2011

- Ces travaux ont donné lieu à une publication avec actes à la conférence EWRL, qui regroupe la communauté internationale d'apprentissage par renforcement.

E. Klein, M. Geist et O. Pietquin. « Batch, Off-policy and Model-free Apprenticeship Learning ». Dans : *Proceedings of the European Workshop on Reinforcement Learning (EWRL 2011)*. Lecture Notes in Computer Science (LNCS). Athens (Greece) : Springer Verlag -

Heidelberg Berlin, sept. 2011

C.2 SCIRL

Vient ensuite le nouvel algorithme d'ARI SCIRL que nous décrivons et étudions au chapitre 5 et qui peut tirer parti des avancées offertes par LSTD-μ. Grâce à une heuristique, il est capable de trouver une récompense à partir uniquement de données expertes.

– Une version préliminaire de ces travaux a donné lieu à une communication sans actes à EWRL.

E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « Structured Classification for Inverse Reinforcement Learning ». Dans : *European Workshop on Reinforcement Learning (EWRL 2012)*. Edinburgh (UK), juin 2012

– Nous avons par la suite présenté nos travaux à la communauté francophone à CAp.

E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « Classification structurée pour l'apprentissage par renforcement inverse ». Dans : *Actes de la Conférence Francophone sur l'Apprentissage Automatique (Cap 2012)*. Nancy, France, mai 2012

– Une version aboutie de ces travaux a été publiée à NIPS.

E. KLEIN, M. GEIST, B. PIOT et O. PIETQUIN. « Inverse Reinforcement Learning through Structured Classification ». Dans : *Advances in Neural Information Processing Systems (NIPS 2012)*. Lake Tahoe (NV, USA), déc. 2012

– Une version étendue de l'article présenté à CAp a été publiée dans la revue francophone RIA.

E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « Classification structurée pour l'apprentissage par renforcement inverse ». Dans : *Revue d'Intelligence Artificielle* (Mai 2013)

C.3 CSI

Notre dernière contribution majeure est l'algorithme CSI (chapitre 6), il s'agit d'un autre algorithme d'ARI un peu plus souple que SCIRL. Les deux algorithmes ont été développés parallèlement, mais les publications concernant CSI sont plus récentes.

– L'algorithme a été présenté aux JFPDA cette année.

E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « Apprentissage par renforcement inverse en cascadant classification et régression ». Dans : *Journées Francophones de Plannification, Décision et Apprentissage (JFPDA)*. 2013

– Il sera également présenté à Prague à ECML/PKDD

E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « A cascaded supervised learning approach to inverse reinforcement learning ». Dans : *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2013)*. Prague (Czech Republic), sept. 2013

– Enfin, une communication étudiant les liens entre classification et ARI, regroupant ces travaux et ceux menés sur SCIRL aura lieu à RLDM2013 :

M. GEIST, E. KLEIN, B. PIOT, Y. GUERMEUR et O. PIETQUIN. « Around inverse reinforcement learning and score-based classification ». Dans : *Reinforcement Learning and Decision Making Meetings*. 2013

L'analyse théorique de CSI est due à un autre doctorant de l'équipe MaLIS, Bilal Piot. Elle précède chronologiquement l'analyse de SCIRL, qui s'en inspire.

C.4 Autres travaux

D'autres travaux, parallèles à ceux présentés ici, n'ont pas donné de résultats satisfaisants. Les résultats préliminaires avaient cependant donné lieu à une publication. L'idée est de réduire la dimension de l'espace d'hypothèses dans lequel on cherche une récompense en exploitant les transformations (translations, multiplications par un facteur positif) qu'il est possible, sur tout PDM, d'appliquer à une récompense sans changer ses politiques optimales. Une modification de l'algorithme du simplexe permet, dans cet espace de dimension réduite, de trouver les récompenses creuses (différentes de zéro en seulement quelques points) expliquant le comportement de l'expert. Malheureusement l'existence de telles solutions n'est pas garantie et la complexité de l'algorithme croît exponentiellement avec le cardinal de l'espace d'état.

E. KLEIN, M. GEIST et O. PIETQUIN. « Reducing the dimentionality of the reward space in the Inverse Reinforcement Learning problem ». Dans : *Proceedings of the IEEE Workshop on Machine Learning Algorithms, Systems and Applications (MLASA 2011)*. Honolulu (USA), déc. 2011

Lors de la 8ème école d'été sur les interfaces multimodales (eNTERFACE'12), nous avons tenté d'appliquer l'ARI à un problème réel et de grande dimension : la commande d'un bras robot par le biais d'une interface cerveau/machine (*Brain Computer Interface*, BCI). En observant un oracle associer la sortie de la BCI (une fois filtrée) à un ordre (avancer, droite, gauche, etc.), le but est de trouver une récompense qui, une fois optimisée, permettrait à un agent de remplacer l'oracle. L'utilisation de l'AR permettrait d'obtenir un agent s'améliorant au fur et à mesure que le nombre d'interactions augmente, et qui s'adapte à l'utilisateur. Les résultats ont été prometteurs, en ce sens que l'algorithme SCIRL (que nous présentons au chapitre 5) a été en mesure de trouver une fonction de récompense. Nous avons cependant manqué de temps pour collecter les données nécessaires à l'optimisation de cette récompense, et n'avons de fait pas pu évaluer la qualité de la récompense fournie par SCIRL.

L. BOUGRAIN, M. DUVINAGE et E. KLEIN. *Inverse reinforcement learning to control a robotic arm using a Brain-Computer Interface*. Rap. tech. eNTERFACE Summer Workshop, 2012

Bibliographie

[ABR64] A. Aizerman, E. M. Braverman et L. Rozoner. « Theoretical foundations of the potential function method in pattern recognition learning ». Dans : *Automation and remote control* 25 (1964), p. 821–837.

[ACN10] P. Abbeel, A. Coates et A. Ng. « Autonomous helicopter aerobatics through apprenticeship learning ». Dans : *International Journal of Robotics Research* 29.13 (2010), p. 1608–1639.

[AN04] P. Abbeel et A. Ng. « Apprenticeship learning via inverse reinforcement learning ». Dans : *International Conference on Machine Learning (ICML)*. ACM. 2004, p. 1.

[BB96] S. Bradtke et A. Barto. « Linear least-squares algorithms for temporal difference learning ». Dans : *Machine Learning* 22.1 (1996), p. 33–57. issn : 0885-6125.

[BDK12] L. Bougrain, M. Duvinage et E. Klein. *Inverse reinforcement learning to control a robotic arm using a Brain-Computer Interface*. Rap. tech. eNTERFACE Summer Workshop, 2012.

[Bel03] R. Bellman. *Dynamic programming*. Dover Pubns, 2003.

[BGV92] B. Boser, I. Guyon et V. Vapnik. « A training algorithm for optimal margin classifiers ». Dans : *Proceedings of the fifth annual workshop on Computational learning theory*. ACM. 1992, p. 144–152.

[BKP11] A. Boularias, J. Kober et Peters. « Relative Entropy Inverse Reinforcement Learning ». Dans : *International Conference on Automated Planning and Scheduling (ICAPS)* 15 (2011), p. 20–27.

[Bon13] R. Bonidal. « Analyse des systèmes discriminants multi-classes à grande marge ». Thèse de doct. Université de Lorraine, 2013.

[BS00] M. Bain et C. Sommut. « A framework for behavioural cloning ». Dans : *Machine Intelligence 15* 15 (2000), p. 103.

[BT04] A. Berlinet et C. Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. T. 3. Kluwer Academic Boston, 2004.

[CGJ96] D. A. Cohn, Z. Ghahramani et M. I. Jordan. « Active learning with statistical models ». Dans : *Journal of Artificial Intelligence Research* (1996).

[CKO01] U. Chajewska, D. Koller et D. Ormoneit. « Learning an agent's utility function by observing behavior ». Dans : *International Conference on Machine Learning (ICML)*. 2001, p. 35–42.

[CV07] S. Chernova et M. Veloso. « Confidence-based policy learning from demonstration using gaussian mixture models ». Dans : *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM. 2007, p. 233.

[CV95] C. Cortes et V. Vapnik. « Support-vector networks ». Dans : *Machine learning* 20.3 (1995), p. 273–297.

[DRP09] M. Deisenroth, C. Rasmussen et J. Peters. « Gaussian process dynamic programming ». Dans : *Neurocomputing* 72.7-9 (2009), p. 1508–1524.

[DT10] K. Dvijotham et E. Todorov. « Inverse Optimal Control with Linearly-Solvable MDPs ». Dans : *Proceedings of the Interntional Conference on Machine Learning*. 2010.

[EGW05] D. Ernst, P. Geurts et L. Wehenkel. « Tree-based batch mode reinforcement learning ». Dans : *Journal of Machine Learning Research* 6 (2005).

[Fri01] J. FRIEDMAN. « Greedy function approximation: a gradient boosting machine ». Dans : *Annals of Statistics* 29.5 (2001), p. 1189–1232.

[Gei+12] M. GEIST, B. SCHERRER, A. LAZARIC et M. GHAVAMZADEH. « A Dantzig Selector Approach to Temporal Difference Learning ». Dans : *International Conference on Machine Learning (ICML)*. (to appear). 2012.

[Gei+13a] M. GEIST, E. KLEIN, Y. GUERMEUR et O. PIETQUIN. *Cascading and Merging Supervised Learning for Inverse Reinforcement Learning*. Rap. tech. 2013.

[Gei+13b] M. GEIST, E. KLEIN, B. PIOT, Y. GUERMEUR et O. PIETQUIN. « Around inverse reinforcement learning and score-based classification ». Dans : *Reinforcement Learning and Decision Making Meetings*. 2013.

[Gha+10] M. GHAVAMZADEH, A. LAZARIC, O.-A. MAILLARD et R. MUNOS. « LSTD with random projections ». Dans : (2010).

[Gor95] G. GORDON. *Stable function approximation in dynamic programming*. Rap. tech. DTIC Document, 1995.

[Gue12] Y. GUERMEUR. « A generic model of multi–class support vector machine ». Dans : *International Journal of Intelligent Information and Database Systems* 6.6 (2012), p. 555–577.

[Had02] J. HADAMARD. « Sur les problemes aux dérivées partielles et leur signification physique ». Dans : *Princeton University Bulletin* 13.49-52 (1902), p. 28.

[How60] R. HOWARD. « Dynamic programming and Markov process ». Dans : (1960).

[JQZ10] Z. JIN, H. QIAN et M. ZHU. « Gaussian processes in inverse reinforcement learning ». Dans : *International Conference on Machine Learning and Cybernetics (ICMLC)*. T. 1. IEEE. 2010, p. 225–230.

[KGP11a] E. KLEIN, M. GEIST et O. PIETQUIN. « Apprentissage par imitation étendu au cas batch, off-policy et sans modèle ». Dans : *Sixièmes Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2011)*. Rouen (France), juin 2011.

[KGP11b] E. KLEIN, M. GEIST et O. PIETQUIN. « Batch, Off-policy and Model-Free Apprenticeship Learning ». Dans : *IJCAI Workshop on Agents Learning Interactively from Human Teachers (ALIHT 2011)*. Barcelona (Spain), juil. 2011.

[KGP11c] E. KLEIN, M. GEIST et O. PIETQUIN. « Batch, Off-policy and Model-free Apprenticeship Learning ». Dans : *Proceedings of the European Workshop on Reinforcement Learning (EWRL 2011)*. Lecture Notes in Computer Science (LNCS). Athens (Greece) : Springer Verlag - Heidelberg Berlin, sept. 2011.

[KGP11d] E. KLEIN, M. GEIST et O. PIETQUIN. « Reducing the dimentionality of the reward space in the Inverse Reinforcement Learning problem ». Dans : *Proceedings of the IEEE Workshop on Machine Learning Algorithms, Systems and Applications (MLASA 2011)*. Honolulu (USA), déc. 2011.

[Kle+12a] E. KLEIN, M. GEIST, B. PIOT et O. PIETQUIN. « Inverse Reinforcement Learning through Structured Classification ». Dans : *Advances in Neural Information Processing Systems (NIPS 2012)*. Lake Tahoe (NV, USA), déc. 2012.

[Kle+12b] E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « Classification structurée pour l'apprentissage par renforcement inverse ». Dans : *Actes de la Conférence Francophone sur l'Apprentissage Automatique (Cap 2012)*. Nancy, France, mai 2012.

[Kle+12c] E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « Structured Classification for Inverse Reinforcement Learning ». Dans : *European Workshop on Reinforcement Learning (EWRL 2012)*. Edinburgh (UK), juin 2012.

[Kle+13a] E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « A cascaded supervised learning approach to inverse reinforcement learning ». Dans : *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2013)*. Prague (Czech Republic), sept. 2013.

[Kle+13b] E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « Apprentissage par renforcement inverse en cascadant classification et régression ». Dans : *Journées Francophones de Plannification, Décision et Apprentissage (JFPDA)*. 2013.

[Kle+13c] E. KLEIN, B. PIOT, M. GEIST et O. PIETQUIN. « Classification structurée pour l'apprentissage par renforcement inverse ». Dans : *Revue d'Intelligence Artificielle* (Mai 2013).

[KN09] J. KOLTER et A. NG. « Regularization and feature selection in least-squares temporal difference learning ». Dans : *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM. 2009, p. 521–528.

[LeC+06] Y. LECUN, U. MULLER, J. BEN, E. COSATTO et B. FLEPP. « Off-road obstacle avoidance through end-to-end learning ». Dans : *Advances in neural information processing systems* 18 (2006), p. 739. ISSN : 1049-5258.

[LGM10] A. LAZARIC, M. GHAVAMZADEH et R. MUNOS. « Finite-sample analysis of LSTD ». Dans : *Proceedings of the 27th International Conference on Machine Learning*. 2010.

[LMM09] M. LOPES, F. MELO et L. MONTESANO. « Active learning for reward estimation in inverse reinforcement learning ». Dans : *Machine Learning and Knowledge Discovery in Databases* (2009), p. 31–46.

[LP03] M. LAGOUDAKIS et R. PARR. « Least-squares policy iteration ». Dans : *The Journal of Machine Learning Research* 4 (2003), p. 1107–1149. ISSN : 1532-4435.

[LPK10] S. LEVINE, Z. POPOVIC et V. KOLTUN. « Feature construction for inverse reinforcement learning ». Dans : *Proc. NIPS*. T. 23. 2010, p. 1342–1350.

[LPK11] S. LEVINE, Z. POPOVIC et V. KOLTUN. « Nonlinear Inverse Reinforcement Learning with Gaussian Processes ». Dans : *Neural Information Processing Systems (NIPS)* (2011).

[Mas+99] L. MASON, J. BAXTER, P. BARTLETT et M. FREAN. « Functional gradient techniques for combining hypotheses ». Dans : *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS* (1999), p. 221–246.

[Mer09] J. MERCER. « Functions of positive and negative type, and their connection with the theory of integral equations ». Dans : *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character* 209 (1909), p. 415–446.

[ML10] F. S. MELO et M. LOPES. « Learning from demonstration using MDP induced metrics ». Dans : *Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, p. 385–401.

[Moo90] A. MOORE. « Efficient Memory-Based Learning for Robot Control ». Thèse de doct. University of Cambridge, 1990.

[Mun07] R. MUNOS. « Performance bounds in L_p norm for approximate value iteration ». Dans : *SIAM journal on control and optimization* 46.2 (2007), p. 541–561.

[NHR99] A. NG, D. HARADA et S. RUSSELL. « Policy invariance under reward transformations: Theory and application to reward shaping ». Dans : *International Conference on Machine Learning (ICML)*. 1999, p. 278–287.

[Nor98] J. R. NORRIS. *Markov chains*. 2008. Cambridge university press, 1998.

[NR00] A. NG et S. RUSSELL. « Algorithms for inverse reinforcement learning ». Dans : *International Conference on Machine Learning (ICML)*. Morgan Kaufmann Publishers Inc. 2000, p. 663–670.

[NS07] G. NEU et C. SZEPESVÁRI. « Apprenticeship learning using inverse reinforcement learning and gradient methods ». Dans : *Conference on Uncertainty in Artificial Intelligence (UAI)*. 2007, p. 295–302.

[NS09] G. NEU et C. SZEPESVÁRI. « Training parsers by inverse reinforcement learning ». Dans : *Machine learning* 77.2 (2009), p. 303–337.

[Pom93] D. A. POMERLEAU. « Knowledge-based training of artificial neural networks for autonomous robot driving ». Dans : *Robot learning*. Springer, 1993, p. 19–43.

[Put94] M. PUTERMAN. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994. ISBN : 0471619779.

[QB11] Q. QIAO et P. BELING. « Inverse reinforcement learning with Gaussian process ». Dans : *American Control Conference (ACC), 2011*. IEEE. 2011, p. 113–118.

[RA07] D. RAMACHANDRAN et E. AMIR. « Bayesian inverse reinforcement learning ». Dans : *Proceedings of the International Joint Conference on Artificial Intelligence* (2007), p. 2586–2591.

[Rat+07] N. RATLIFF, D. BRADLEY, J. BAGNELL et J. CHESTNUTT. « Boosting structured prediction for imitation learning ». Dans : *Advances in Neural Information Processing Systems* 19 (2007), p. 1153.

[RB10] S. ROSS et J. A. BAGNELL. « Efficient reductions for imitation learning ». Dans : (2010).

[RBS07] N. RATLIFF, J. BAGNELL et S. SRINIVASA. « Imitation learning for locomotion and manipulation ». Dans : *International Conference on Humanoid Robots*. IEEE. 2007, p. 392–397.

[RBZ06] N. RATLIFF, J. BAGNELL et M. ZINKEVICH. « Maximum margin planning ». Dans : *International Conference on Machine Learning (ICML)*. ACM. 2006, p. 736.

[RGB10] S. ROSS, G. J. GORDON et J. A. BAGNELL. « A reduction of imitation learning and structured prediction to no-regret online learning ». Dans : *arXiv preprint arXiv:1011.0686* (2010).

[Rie05] M. RIEDMILLER. « Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method ». Dans : *Machine Learning: ECML 2005* (2005), p. 317–328.

[RK04] C. RASMUSSEN et M. KUSS. « Gaussian processes in reinforcement learning ». Dans : *Advances in Neural Information Processing Systems* 16.1 (2004), p. 751–759.

[RSB09] N. RATLIFF, D. SILVER et J. BAGNELL. « Learning to search: Functional gradient techniques for imitation learning ». Dans : *Autonomous Robots* 27.1 (2009), p. 25–53.

[Rus98] S. RUSSELL. « Learning agents for uncertain environments (extended abstract) ». Dans : *Annual Conference on Computational Learning Theory*. ACM. 1998, p. 103.

[Sam+92] C. SAMMUT, S. HURST, D. KEDZIER et D. MICHIE. « Learning to fly ». Dans : *Proceedings of the ninth international workshop on Machine learning*. 1992, p. 385–393.

[SB98] R. SUTTON et A. BARTO. *Reinforcement learning*. MIT Press, 1998.

[SBS08] U. SYED, M. BOWLING et R. SCHAPIRE. « Apprenticeship learning using linear programming ». Dans : *International Conference on Machine Learning (ICML)*. ACM. 2008, p. 1032–1039.

[SC98] A. SRINIVASAN et R. CAMACHO. « Inductive Logic Programming Applied to an Area of Flight Control ». Dans : *Machine Intelligence* 15 (1998).

[SG11] B. SCHERRER et M. GEIST. « Recursive Least-Squares Learning with Eligibility Traces ». Dans : *Proceedings of the European Workshop on Machine Learning (EWRL 2011)*. Lecture Notes in Computer Science (LNCS). Athens (Greece) : Springer Verlag - Heidelberg Berlin, sept. 2011, 12 pages.

[SL91] S. R. SAFAVIAN et D. LANDGREBE. « A survey of decision tree classifier methodology ». Dans : *Systems, Man and Cybernetics, IEEE Transactions on* 21.3 (1991), p. 660–674.

[SND06] J. SAUNDERS, C. L. NEHANIV et K. DAUTENHAHN. « Teaching robots by moulding behavior and scaffolding the environment ». Dans : *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. ACM. 2006, p. 118–125.

[SS08] U. SYED et R. SCHAPIRE. « A game-theoretic approach to apprenticeship learning ». Dans : *Advances in neural information processing systems* 20 (2008), p. 1449–1456.

[SS97] G. SHIRAZ et C. SAMMUT. « Combining knowledge acquisition and machine learning to control dynamic systems ». Dans : *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*. T. 15. LAWRENCE ERLBAUM ASSOCIATES LTD. 1997, p. 908–913.

[Sti95] D. STIRLING. « CHURPs: Compressed Heuristic Universal Reaction Planners ». Thèse de doct. Ph. D. Thesis, University of Sydney, 1995.

[Tas+05] B. TASKAR, V. CHATALBASHEV, D. KOLLER et C. GUESTRIN. « Learning structured prediction models: A large margin approach ». Dans : *Proceedings of the 22nd international conference on Machine learning*. ACM. 2005, p. 903.

[Wat89] C. WATKINS. « Learning from delayed rewards (Ph. D. dissertation) ». Dans : *Kings College, Cambridge, England* (1989).

[Zie+08] B. ZIEBART, A. MAAS, J. BAGNELL et A. DEY. « Maximum entropy inverse reinforcement learning ». Dans : *AAAI Conference on Artificial Intelligence (AAAI)*. 2008, p. 1433–1438.